TOWARDS AUTOMATICALLY EVALUATING SECURITY RISKS AND PROVIDING CYBER INTELLIGENCE

A Thesis
Presented to
The Academic Faculty

by

Xiaojing Liao

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
Augest 2017

TOWARDS AUTOMATICALLY EVALUATING SECURITY RISKS AND PROVIDING CYBER INTELLIGENCE

Approved by:

Professor Raheem Beyah,
Committee Chair
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Raheem Beyah, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor John Copeland
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Henry Owen
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Vijay K. Madisetti
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor XiaoFeng Wang
School of Informatics and Computing
Indiana University Bloomington

Date Approved: June 12th 2017

To my daughter.

Lorie Xing,

My heroic dream when exhausted.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my deepest appreciation to my advisor, Dr. Raheem A. Beyah. For the past five years, he offers me invaluable guidance, inspiration, and support. His generous help made this dissertation possible. I am proud to be a member of the Communications Assurance and Performance (CAP) at the Georgia Institute of Technology. I also want to thank Dr. XiaoFeng Wang. It has been with his constant encouragement and priceless mentorship that has helped me to grow within my role as a researcher.

Also, I would like to extend my gratitude to my committee members, Dr. Henry Owen, Dr. John Copeland and Dr. Vijay K. Madisetti for generously serving as my committee members. I also want to thank them for their brilliant comments and suggestions.

Special thanks go to my group members, fellow students, friends, and colleagues, Dr. Selcuk Uluagac, Dr. Shouling Ji, David Formby, Shukun Yang, Weiqing Li, Qingchen Gu, Dr. Sumayah Alrwais, Dr. Zhou Li, Dr. Damon Mccoy, Dr. Elaine Shi, Dr. Shuang Hao, Dr. Chang Liu, Dr. Hanxin Duan, Kan Yuan, Eihal Alowaisheq, Nan Zhang, Xiaolong Bai, Dr. Kai Chen, Xianghang Mi, Peng Wang. They provided me invaluable suggestions and helped for my study and research.

Last but not least, I would like to thank my family for their continuous support, understanding, and help. They are there whenever I need them. This dissertation is dedicated to them. I would like to convey a sincere appreciation to my husband, Luyi Xing, who provided me with support throughout my Ph.D. It has truly been with his love, unwavering support and encouragement that has helped me to finalize this dissertation.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

AUC Area Under Curve, p. 95.

AV anti-virus, p. 96.

Bar Bad Repository, p. 81.

BP Blog Preprocessor, p. 25.

BS Blog Scraper, p. 25.

BUS Bucket Usage Similarity, p. 89.

C&C Command and Control, p. 23.

CDN Content Distribution Network, p. 80.

CI Composite Importance, p. 29.

CMS Content Management System, p. 90.

CR Connection Ratio, p. 89.

CTI Cyber Threat Intelligence, p. 19.

CVE Common Vulnerabilities and Exposures, p. 15.

DG Dependency Graph, p. 22.

DOM Document Object Model, p. 58.

FDR False Discovery Rate, p. 95.

FQDN Fully Qualified Domain Names, p. xi.

gTLD Generic Top-level Domains, p. 116.

IBT Irrelevant Bad Terms, p. 12.

IG IOC generator, p. 26.

IOC Indicator of Compromise, p. x.

NER Name Entity Recognition, p. 2.

NLP Natural Language Processing, p. 7.

POS Parts-of-Speech, p. 9.

PUP Potentially Unwanted Programs, p. 84.

RC Relation Checker, p. 26.

RCP Relevant-content Picker, p. 25.

RE Relation Extraction, p. 15.

ROC Receiver Operating Characteristics, p. 95.

S3 Simple Storage Service, p. 80.

SEISE Semantic Inconsistency Search, p. xi.

SEO search engine optimization, p. 16.

sTLD Sponsored Top-level Domains, p. 12.

SVM Support Vector Machine, p. 15.

UCG User-generated Content, p. 29.

SUMMARY

The cyber threat landscape is quickly changing and it is of vital importance to stay abreast of emerging threats and to proactively work to improve security. At the same time, piecing together a complete landscape of attacks by identifying the strategies and capabilities of the adversaries requires establishing links among individual observations. Also, defending against these attacks requires automatically generated semantics-aware policies to complement manual analysis. While using data processing techniques and semantic-aware inspection to address security problems is a promising approach to evaluate security risks and to provide cyber intelligence, there exists a gap between the security ontology and generic data processing techniques primitives needed for such an approach. This gap tends to be domain-sensitive, language-specific, and computationally intensive which further complicates the use of such an approach.

In this dissertation, data processing techniques and semantic-aware inspection were applied in three componemts of the modern security system: cyber threat gathering, cyber threat analysis, and cyber threat detection. From the developed modern security system, three techniques are suggested as contributions: (1) we present an innovation solution for fully automated IOC extraction. It solved a name entity recognition problem for the security domain through an innovative graph mining technique. Running on 71,000 articles collected from 45 leading technical blogs, this new approach demonstrates a remarkable performance: it generated 900K OpenIOC items with a precision of 95% and a coverage over 90%, which is way beyond what the state-of-the-art natual language processing techniques and commercial indicator of compromise (IOC) tools can achieve, at a speed of thousands of articles per hour.

Further, by correlating the IOCs mined from the articles published over a 13-year span, our study sheds new light on the links across hundreds of seemingly unrelated attack instances, particularly their shared infrastructure resources, as well as the impacts of such open-source threat intelligence on security protection and evolution of attack strategies. (2) we analyzed emerging cyber threats on cloud platform using big data analytics techniques. We identified a set of collective features, which uniquely characterize malicious cloud repositories. These features were utilized to build a scanner that detected over 600 malicious cloud repositories on leading cloud platforms like Amazon, Google, and 150K sites, including popular ones like `groupon.com`, using them. Highlights of our study include the pivotal roles played by these repositories on malicious infrastructures and other important discoveries include how the adversary exploited legitimate cloud repositories and why the adversary uses bad cloud repositories in the first place, which has never been reported. These findings bring such malicious services to the spotlight and contribute to a better understanding and ultimate elimination on this new threat. (3) we developed a semantic-based technique, called *Semantic Inconsistency Search* (SEISE), for efficient and accurate detection of the promotional injections on sponsored top-level domains (sTLD) with explicit semantic meanings. Running on 403 sTLDs with an initial 30 seed irrelevnt bad terms, SEISE analyzed 100K fully qualified domain names (FQDN), and along the way automatically gathered nearly 600 IBTs. In the end, our approach detected 11K infected FQDN with a false detection rate of 1.5% and over 90% coverage. Our findings further bring to light the stunning impacts of such promotional attacks, which compromise approximately 3% of the FQDNs in the `.edu`, `.gov` domains and over one thousand `gov.cn` domains, including those of leading universities such as `stanford.edu`, `mit.edu`, `princeton.edu`, `havard.edu` and government institutes such as `nsf.gov` and `nih.gov`. We further demonstrate the potential to extend our current technique to protect generic domains such as `.com` and `.org`.

CHAPTER I

INTRODUCTION

1.1 Motivation

Every day, about 20 new cyber vulnerabilities are released and reported [5]. It comes up with around 110,000 new cyber attacks per hour. In recent years, cyber security becomes a more complex and multifaceted problem because of the fast-evolving threat landscape. The threat environment is quickly changing and it is of vital importance to stay abreast of emerging threats and to proactively work to improve security. Traditional methods for cyber security system design focus on understanding and addressing each of vulnerabilities, weaknesses, and configurations separately, which are insufficient to defend against the quickly increasing cyber threats. Effective defense against current and future threats requires the addition of a balancing, outward focus, on understanding the adversarys behavior, capability, and intent.

However, piecing together a complete landscape of attacks by identifying the strategies and capabilities of the adversaries requires establishing links among individual observations through a large amount of cyber threat data analysis. Also, defending against these current or future attacks requires automatically generated semantic-aware cyber intelligence to complement manual analysis. For example, with cyber intelligence, cyber defenders can understand and characterize things like what sort of attack actions have occurred and are likely to occur; how can these actions be detected and recognized; how can they be mitigated; who are the relevant threat actors; what are they trying to achieve; what are their capabilities, in the form of tactics, techniques, and procedures they have leveraged over time and are likely to leverage in the future; what sort of vulnerabilities, misconfigurations, or weaknesses

they are likely to target; what actions have they taken in the past; etc.

While using data-oriented design and semantic-aware techniques to address security problems is a promising approach to evaluate security risks and to provide cyber intelligence, there exists a gap between the security ontology and general data processing techniques (such as machine learning techniques, natural language processing techniques) needed for such an approach. This gap tends to be domain-sensitive, language-specific, and computationally intensive which further complicates the use of such an approach. For example, when adapting machine learning techniques in security detection tools, robust security domain features should be carefully selected to consider the potential evasion by the adversary: for example, an adversary can attempt to fool classifiers by purposely modifying their behavior. Another challenge to adopt natural language processing technique in security application is that we found that the Stanford name entity recognition (NER) tool cannot be directly applied to recover indicators of compromises (IOC, e.g., malware signatures, botnet IPs) from online blogs, since it resulted in a precision (around 70%) and recall (below 50%) far below what is expected for a security application (e.g., signatures for an Anti-Virus scanner) [81]. Additionally, cyber attacks are global and could require an investigation across a large amount of data in different languages.

In this dissertation, we applied data-driven security design and semantic-aware techniques in a modern security system. We have created a set of methods in each component of a modern security system that enable the system to evaluate security risks and provide cyber intelligence automatically.

1.2 Modern Security System

Cyber Threat Gathering. To enable an organization to determine its responses at the strategic, operational and tactical levels [99], the first step to designing a modern security system is cyber threat gathering. Cyber threat gathering is to collect

and identify information that details the current and emerging security threats. An example output of cyber threat gathering is Indicator-of-compromise (IOC), which elaborates the forensic artifacts of an attack and can, therefore, be used to analyze the attack once it happens or counter it during its execution. An IOC includes not only individual data fingerprints involved in a specific attack, such as an attack domain, the MD5 of the malware delivered, but also the context of the attack and an analysis of the adversary's behavior, like the type of the attack or the specific technique deployed (e.g., change to the Windows Registry). To find out such information, cyber threat gathering includes identification of the adversary's tools, techniques and attack procedures, which, together with the fingerprints, helps an organization's security team to understand their security posture, detect early signs of threats and continuously improve their security controls.

The sources for cyber threat gathering can be closed, e.g., a corporation's internal network traces, or public, such as technical blogs or online forums. Examples of public sources include the blogs of AlienVault, FireEye, Malwarebytes, and Webroot. With the surge of cyber attacks in recent years, a large number of attack artifacts have emerged, which has been extensively reported by the public online sources and aggressively collected by different organizations. To bootstrap our research on automatically cyber threat gathering, we reached out to a security company and obtained a list of 45 blogs which were operated by renowned organizations and practitioners. These blogs altogether covered major security incidents in the world.

We monitored these 45 blogs and were able to download as many as 71,000 articles. Also noteworthy is that the number of cyber threat-related articles there continued to grow in the past 13 years (2003/01 to 2016/04), from 20 to 1,000 per month, with an increased rate of 500% (see Figure 5(a)). While the volume of articles we have collected is substantial, it only constitutes a small piece of the IOC landscape. Rapidly collecting and sharing such information, and deploying it to various security systems

is the key to quickly detecting, responding and containing different kinds of security threats organizations face, which requires the descriptions of the cyber threat to be standardized and machine-digestible. To this end, various cyber threat information sharing frameworks have been proposed, including OpenIOC [18], STIX [17] and yara [22], with each format easily convertible to the others.

OpenIOC is an extensible XML schema created by Mandiant to record technical characteristics that identify a known threat, an attacker's methodology, or other evidence of a compromise. As an example, Figure 1 shows how to use the OpenIOC format to describe a family of POS malware, *Alina* [69]. Such a description includes two components, a header and a definition. The header part has a summary of the attack (under the `description` tag) and the source of the information (under `authored_by` and `authored_date`). The definition contains a set of indicator items (under `Indicator Item`), each representing an IOC (`Content`) and its context (`Context`). Two such items are illustrated in Figure 1. In each of them, the `document` attribute under `Context` gives the main category of an IOC (e.g., process, event, file, registry, etc.) and `search` elaborates on its subcategory using an *iocterm*, essentially a concatenation of common terminologies (e.g., process, name, etc.) security professionals use to describe the IOC. The OpenIOC standard provides 600 such iocterms, covering various types of artifacts left by different attacks, such as cookie history, DNS entry, Email information, Hook items and others. The examples of iocterms related to IP, hash and datetime are shown in Table 1. For example, In the case of IP, its iocterms describe 15 different contexts, such as spammer's IP (`Email/ReceivedFromIP`), IP in malicious infrastructure (`RouteEntryItem/Destination`), or C&C attack's remote IP (`ProcessItem/PortList/PortItem/remoteIP`).

In Figure 1, the two indicator items record the registry change made by the Alina malware: the first shows the location where the change happens, i.e., the path `Windows\CurrentVersion\Run` with the context `RegistryItem/KeyPath`, and the

Table 1: Examples of iocterms and their corresponding general type and context terms.

General type	iocterms	context terms
IP	Email/ReceivedFromIP, PortItem/remoteIP	email, IP, received, remote, port
Hash	ServiceItem/serviceDLL-sha1sum, FileItem/Md5sum	service, dll, sha1, md5
Date-time	Email/Date, EventLogItem/genTime	email, date, event, log, generation

second identifies the name of the code dropped there, i.e., `ALINA`.

Cyber Threat Analysis. In cyber threat analysis, cyber threats were modeled by analyzing cyber threat information. Cyber threat analysis seeks to understand the nature of relevant threats, identify them, and fully characterize them such that all of the relevant knowledge of the threat can be fully expressed and evolved over time. This relevant knowledge includes threat-related actions, behaviors, capabilities, intents, attributed actors, etc. From this understanding and characterization, the analyst may then specify relevant threat indicator patterns, suggest courses of action for threat response activities, and/or share the information with other trusted parties. For example, in the case of a potential phishing attack, a cyber threat analyst may analyze and evaluate a suspected phishing email, analyze any email attachments and links to determine if they are malicious, determine if the email was sent to others, assess commonality of who/what is being targeted in the phishing attack, determine whether malicious attachments were opened or links followed, and keep a record of all analysis performed.

In recent years, the popularity of cloud platform has brought with new challenges to cyber threat analysis: incidents of developing new blackhat techniques for cloud platforms or utilizing reputable cloud platforms to serve their malicious online activities have been reported. Cloud hosting is a type of infrastructure as a service, which is rented by the cloud user to host her web assets (e.g., HTML, JavaScript, CSS, and

image files). A key feature of cloud hosting is built-in site publishing [59], where the web assets in the cloud repositories can be served directly to users via file names in a relative path in the cloud repositories (i.e., *cloud URL*). For instance, JavaScript files hosted in the cloud repositories can be directly run in the browser. Also, the pay-as-you-go hosting is well received as an economic and flexible computing solution. As an example, Google Drive today offers a free web hosting service with 15GB of storage, and an additional 100GB for $1.99/month and GoDaddy's web hosting starts at $1/month for 100GB.

Although there have been indications of cloud hosting misuse, analyzing the cyber threat in cloud platforms is challenging: the service providers are bound by their privacy commitments and ethical concerns. Hence, they tend to avoid inspecting the content of their customers repositories in the absence of proper consent. Even when the providers are willing to do so, determining whether a repository involves malicious content is by no means trivial: nuts and bolts for malicious activities could appear perfectly innocent before they are assembled into an attack machine; examples include image files for Spam and Phishing. In our research, we analyze the critical and emerging cyber threats that the adversary who tries to use cloud repositories on legitimate cloud platforms as service repositories for illicit activities. For this purpose, the attacker could build her malicious cloud repositories or compromise legitimate ones, and store various attack vectors there, including Spam, Phishing, malware, click-hijacking and others. These cloud repositories are connected to front-end websites, which could be malicious, compromised or legitimate ones contaminated only by the malicious cloud repositories.

Cyber Threat Detection. After modeling cyber threats, a cyber threat detection system is designed to detect cyber threats automatically and accurately. At a high level, a cyber threat detection system must be efficient, accurate, and adaptive to an

environment that can drastically change over time. Considering efficiency, it is important the detection system scale favorably with large, possibly infinite datasets. The system should thus use an online detection algorithm for detecting from a streaming data source. Also, cyber threat detection system should be accurate. Both false positives and false negatives need to be taken into account to evaluate the performance. Furthermore, both the cyber content and the threats attackers vary drastically over time. As new exploits are discovered, or old vulnerabilities are being patched, the cyber threats change over time. The detection system should thus be able to learn the evolution of these threats.

Nowadays, most online information is generated and consumed by humans, and many of todays cyber threats directly target individual humans. A prominent example is Internet fraud, in which fraudulent content carried by web pages, phishing emails, or user interfaces of malicious apps often cannot be captured without understanding its semantic meaning. The need for such deep understanding has prompted recent efforts on semantics-aware cyber threat detection, which leverages large-scale semantic processing of text content to create new security defenses. For example, if you Googled "`site:stanford.edu pharmacy`" a few months ago, you would have found Stanford University web pages displaying advertisements (ads) for cheap Viagra. This was the result of a *promotional infection* for underground advertising [80]. Such infections affected a large number of `.edu` and `.gov` domains, including `mit.edu`, `princeton.edu`, `havard.edu`, `state.gov`, `nih.gov`, and even `nsf.gov` (for link Spam) [80]. The infections remained unnoticed for months, owing to their similarity to legitimate ad injections. We detected them only with a *semantics-aware inspection* [80], using Natural Language Processing (NLP) techniques to expose the discrepancy between ads and typical content for such sponsored top-level domains (sTLDs).

The challenge of semantics-aware cyber threat detection comes from the diversity of security domains, ranging from fraud to malware to threat intelligence to privacy,

each with different objectives and datasets. This diversity makes it hard to set a unified target and share research findings and new techniques across domains. Further complicating the situation is the gap between the requirements for security protection and the capabilities of generic NLP primitives, which tend to be domain-sensitive, language-specific, and computationally intensive. For example, we found that the Stanford NER cannot be directly applied to recover indicators of compromises (IOC, e.g., malware signatures, botnet IPs) from online blogs, since it resulted in a precision (around 70%) and recall (below 50%) far below what is expected for a security application (e.g., signatures for an Anti-Virus scanner) [81]. Additionally, cyber attacks are global and could require an investigation across a large amount of data in different languages. A multilingual and scalable design is demanded such an investigation.

1.3 Data-driven Security Design

To automatically evaluate security and provide cyber intelligence, we leveraged a set of data processing techniques (such as machine learning and natural language processing) in modern security system design. Also, due to the unique characteristics of this open problem in the security domain, the general data processing techniques cannot achieve good performance for our tasks, and therefore we also applied security domain knowledge to optimize data processing techniques to achieve better performance in the security application. These data processing techniques are briefly introduced here.

Dependency parsing. Dependency parsing is an NLP technique for describing grammatical relations between words in a sentence. Such relations include a direct object, determinant, noun compound modifier and others. Also, the content of a relative clause is further analyzed to identify the dependencies between the words it includes. For example, the sentence in Figure 2 is parsed into dependency relations, such as the determinant relation between "trojan" and "the", `det(Trojan-2,`

`the-1)` (where the number shows the position of the word in the sentence), and the nominal subject relation between "trojan" and "download", `nsubj(downloads-3, Trojan-2)`. Each of the formulae represents a binary relation between a *governor* (the first term) and a *dependent* (the second one).

Such dependency relations within a sentence form a directed and weighted graph (V, E, W), where each token is a vertex in V , and the relation between them is represented by a set of edges in E. Each arc connecting two tokens can also be assigned a *weight* W to differentiate the relation between them from others. Figure 2 further illustrates the dependency graph for the example sentence. The state-of-the-art dependency parser (e.g., Stanford's typed dependencies system [39]) can achieve a 92.2% unlabeled attachment score in discovering the grammatical relations in a sentence. In our research, we utilized the relations discovered by the parser to capture the semantic links between context terms and an IOC token. For example, the dependency of `clickme.zip` on "attachments" in the sentence "all e-mails collected have had attachments clickme.zip" reveals a compound relation between the terms (the content of the "attachment" is `clickme.zip`), which falls in line with the descriptions typically explaining the security issues related to email attachments.

Content term extraction. Another set of techniques extensively utilized in information extraction is content term extraction, which automatically determines important terms within a given piece of text. It includes parts-of-speech (POS) tagging that labels a word corresponding to a particular part of speech (such as nouns and verbs), and phrase parsing that divides sentences into phrases logically belonging together. Specifically, after parsing phrases from the given content, a POS tagger labels its terminological candidates, such as syntactically plausible terminological noun phrases. Then, these candidates are analyzed using statistical approaches (e.g., point-wise mutual information) to find out important terms.

Graph mining. Our approach tailors graph mining techniques to analyze the dependency graphs constructed from sentences of interest. Graph mining is a structural data mining problem with the purpose of identifying distinguishing characteristics (such as common subgraph) of graphs. The problem can be stated as follows. Given a dataset of graphs $G_i \in D$, with $i = 1...N$, each graph $G_i = (V_i, E_i)$ is a collection of vertices $V_i = \{v_{i1}, \cdots, v_{in}\}$ and edges $E_i = \{(v_a, v_b) | v_a, v_b \in V_i\}$. G_i may also have labels on its nodes and edges, which are drawn from a common label set of the whole dataset D. Also, each graph G_i is in a *class* with a label $c_i \in C$. The goal of the *graph classification* problem is to learn a model $f : D \rightarrow C$ that predicts the class label for any graph, that is, classifying the graph to a class based on its similarity to other graphs as measured by various *graph kernel* methods, such as direct product kernel [54] and subtree kernel [95]. In our research, we utilize graph mining to determine whether a sentence involving context terms and an IOC token indeed contains IOC.

Word embedding (skip-gram model). A word embedding $W : words \rightarrow V^n$ is a parameterized function mapping words to high-dimensional vectors (200 to 500 dimensions), e.g., $W(`education') = (0.2, -0.4, 0.7, ...)$, to represent the word's relation with other words. Such a mapping can be done in different ways, e.g., using the continual bag-of-words model and the skip-gram technique to analyze the context in which the words show up. Such a vector representation ensures that synonyms are given similar vectors and antonyms are mapped to dissimilar vectors. Also interestingly, the vector representations fit well with our intuition about the semantic relations between words: e.g., the vectors for the words 'queen', 'king', 'man' and 'woman' have the following relation: $v_{queen} - v_{woman} + v_{man} \approx v_{king}$. In our research, we utilized the vectors to compare the semantics meanings of different words, by measuring the cosine distance between the vectors. For example, using Wikipedia pages as a training set (for the context of individual words), our approach automatically

identified the words semantically-close to 'casino', such as 'gambling' (with a cosine distance 0.35), 'vegas' (0.46) and 'blackjack' (0.48).

1.4 Contribution

The contributions of the dissertations are summarized below:

First, we present an innovation solution for fully automated IOC extraction. Our approach is based on the observation that the IOCs in technical articles are often described in a predictable way: being connected to a set of context terms (e.g., "download") through stable grammatical relations. Leveraging this observation, iACE is designed to automatically locate a putative IOC token (e.g., a zip file) and its context (e.g., "malware", "download") within the sentences in a technical article, and further analyze their relations through a novel application of graph mining techniques. Once the grammatical connection between the tokens is found to be in line with the way that the IOC is commonly presented, these tokens are extracted to generate an OpenIOC item that describes not only the indicator (e.g., a malicious zip file) but also its context (e.g., download from an external source). Running on 71,000 articles collected from 45 leading technical blogs, this new approach demonstrates a remarkable performance: it generated 900K OpenIOC items with a precision of 95% and a coverage over 90%, which is way beyond what the state-of-the-art NLP technique and industry IOC tool can achieve, at a speed of thousands of articles per hour. Further, by correlating the IOCs mined from the articles published over a 13-year span, our study sheds new light on the links across hundreds of seemingly unrelated attack instances, particularly their shared infrastructure resources, as well as the impacts of such open-source threat intelligence on security protection and evolution of attack strategies.

Second, we took the first step toward understanding and detecting this emerging threat. Using a small set of "seeds" (i.e., confirmed malicious cloud repositories),

we identified a set of collective features from the websites they serve (e.g., attempts to hide malicious cloud repositories), which uniquely characterize malicious cloud repositories. These features were utilized to build a scanner that detected over 600 malicious cloud repositories on leading cloud platforms like Amazon, Google, and 150K sites, including popular ones like `groupon.com`, using them. Highlights of our study include the pivotal roles played by these repositories on malicious infrastructures, and other important discoveries include how the adversary exploited legitimate cloud repositories and why the adversary uses Bars in the first place that has never been reported. These findings bring such malicious services to the spotlight and contribute to a better understanding and ultimately eliminating this new threat.

Third, we developed a semantic-based technique, called *Semantic Inconsistency Search* (SEISE), for efficient and accurate detection of the promotional injections on sponsored top-level domains (sTLD) with explicit semantic meanings. Our approach utilizes NLP to identify the bad terms (those related to illicit activities like fake drug selling, etc.) most irrelevant to an sTLD's semantics. These terms, which we call *irrelevant bad terms* (IBTs), are used to query search engines under the sTLD for suspicious domains. Through a semantic analysis on the results page returned by the search engines, SEISE can detect those truly infected sites and automatically collect new IBTs from the titles/URLs/snippets of their search result items for finding new infections. Running on 403 sTLDs with an initial 30 seed IBTs, SEISE analyzed 100K FQDN, and along the way automatically gathered nearly 600 IBTs. In the end, our approach detected 11K infected FQDN with a false detection rate of 1.5% and over 90% coverage. Our study shows that by effective detection of infected sTLDs, the bar to promotion infections can be substantially raised, since other non-sTLD vulnerable domains typically have much lower Alexa ranks and are therefore much less attractive for underground advertising. Our findings further bring to light the stunning impacts of such promotional attacks, which compromise FQDNs under 3%

of .edu, .gov domains and over one thousand gov.cn domains, including those of leading universities such as stanford.edu, mit.edu, princeton.edu, havard.edu and government institutes such as nsf.gov and nih.gov. We further demonstrate the potential to extend our current technique to protect generic domains such as .com and .org.

1.5 Organization

The rest of the dissertation is organized as follows: Chapter 2 provides the background information and literature review; Chapter 3 presents an automatic cyber threat gathering system; Chapter 5 and Chapter 6 show cyber threat analysis of the emerging online crime in the cloud platform; Chapter 7 presents an effcient and accurate promotional infection detection system; Chapter 8 concludes the dissertation.

CHAPTER II

RELATED WORK

This chapter provides the background literature study in data-driven and semantic-aware modern security system design. In particular, the first section presents the recent studies in cyber threat gathering, including threat intelligence exchange, and natural language processing techniques for security system design. The next section discusses the related work of cyber threat analysis for blackhat search engine optimization. The third section provides a literature review of injected websites detection.

2.1 Cyber Threat Gathering

Threat intelligence exchange. To help the organizations and security community defend against the fast-evolving cyber attacks, there have been great efforts on threat intelligence sharing. Facebook ThreatExchange [52] and Defense Industrial Base voluntary information sharing program (dibnet) [6] are platforms developed for exchanging IOCs between certified participants. Meanwhile, AlienVault OTX [30], OpenIOC DB [1] and IOC Bucket [67] are established to share public (unclassified) IOCs. Regardless of the type of platform, public sources like blogs still contribute a big portion of IOCs. Our approach, *iACE*, will contribute to the establishment of a fast tunnel between these public sources and exchange platforms and help the participated organizations receive IOC updates timely. As far as we know, AlienVault [30] and Recorded Future [26] are the only two IOC providers that support automatic IOC extraction. Even though Recorded Future (which does not provide public services) utilized NER techniques [96], both tools are simply looking for IOC entities without linking them to attack context, and therefore cannot generate machine-readable OpenIOC items. Instead, *iACE* customized graph mining techniques to capture the

relation between entities, which produces high-quality IOCs and their attack context with high accuracy. IOC extraction from other sources were also studied recently. Catakoglu et al. [38] demonstrated a framework to extract external components (web IOCs) from web pages served in honeypots, which compared with our approach, is more in line with the work on automatic signature generation. Sabottke et al. [100] developed a warning system to alert the user of the ongoing attacks reported by tweets (i.e., looking for the tweets with Common Vulnerabilities and Exposures (CVE)). This is different from our work, which aims at generating machine-readable IOCs from attack reports.

NER/RE. NER today mainly relies on a sequence of words to identify the presence of pre-defined entities (such as PERSON, LOCATION). For example, Stanford NER [53] utilize a Hidden Markov model to find the most likely sequence of entities from unstructured text. Other examples include Illinois NER [41] (based on supervised learning) and LIPI [29] (based on n-gram character language models, etc.). When it comes to Relation Extraction (RE), today's approaches use the tree kernel with support vector machine (SVM) [47], heuristic matches with self-supervised learning [122], open pattern templates [102] and other techniques to detect specific relations between two known entities. By comparison, iACE leverages the unique features of IOC-related articles, using the relation detection to help identify true IOCs and their context. This combines both NER and RE steps together, which has never been done before. Our customized graph mining algorithm also enriches the RE techniques.

NLP for security and privacy. Compared with its application in other areas (e.g., bioinformatics), NLP has only been recently used for security and privacy research. Prior work utilized NLP for analyzing web privacy policies (by extracting its key terms) [127], generating privacy policies for Android apps [126], analyzing app

descriptions to infer required permissions by Android apps [94, 92], detecting compromised websites [80] and identifying sensitive user input from apps [65, 88]. Our work showcases a new application of NLP, demonstrating that innovative NLP techniques need to be developed to address real-world security challenges.

2.2 Cyber Threat Analysis

Study on blackhat SEO. Among the malicious activities performed by a promotional infection is blackhat search engine optimization (SEO, also referred to webspam), which has also been intensively studied. For instance, Wang et al. investigated the longitudinal operations of SEO campaigns by infiltrating an SEO botnet [119]. Leontiadis et al. conducted a long-term study using 5 million search results covering nearly 4 years to investigate the evolution of search engine poisoning [74]. Also, Wang et al. examined the effectiveness of the interventions against the SEO abuse for counterfeit luxury goods [117]. Moore et al. studied the trending terms used in search-engine manipulation [85]. Also, Leontiadis et al. observed `.edu` sites that were compromised for search redirection attack in illicit online prescription drug trade, and briefly discussed their lifetime and volume [71]. In our paper, we conduct a more comprehensive measurement on 403 sTLD, and multiple illicit practices beside drug trade were involved.

Affiliate Program. Recent studies have investigated spam affiliates that send spam through their own email delivery infrastructure and receive a cut of the final revenue for every purchase they bring to the spam-advertised sites [40][75][101]. McCoy et al. analyzed customer demand and overhead in the spam cost model by using transaction logs of pharmaceutical affiliate programs [82]. Caballero et al. infiltrated malware distribution affiliates and measured the pay-per-install market [37]. We supplement prior research by *characterizing the affiliates and affiliate networks abusing cloud web hosting services*.

Search poisoning. Miscreants use search poisoning attacks to falsely increase the rank for their web sites. Previous studies have examined the lexical patterns of the page content [90][114], the hyper-link structure from site to site [62][121], or the combination of the aforementioned features as well as network-level features [116]. deSEO used URL signatures to identify malicious SEO campaigns of fake pages hosted on compromised web servers [68]. Leontiadis et al. did an in-depth analysis of search poisoning attacks which redirected traffic to online pharmacies and found that the conversion rate was higher than email spam [72]. They further used data collected over four years to investigate the evolution of search engine poisoning, which showed that search poisoning attacks have steadily grown. A potential bottleneck is the relatively small set of traffic redirectors was highlighted by Leontiadis et al. [73]. In this paper, we focus on long-tail search-result manipulation based on cloud-hosted pages.

2.3 Cyber Threat Detection

Detection of injected sites. How to detect injection of malicious content has been studied for long. Techniques have been developed to analyze web content, redirection chains and URL pattern. Examples of the content-based detection include a DOM-based clustering systems for monitoring Scam websites [49], and a system monitoring the evolution of web content, called *Delta* [34], which keeps track of the content and structure modifications across different versions of a website, and identifies an infection using signatures generated from such modifications. More recently, Soska et al. works on detecting new attack trends instead of the attacks themselves [110]. Their proposed system leverages the features from web traffic, file system and page content, and is able to predict whether currently benign websites will be compromised in the near future. Borgolte et al. introduces *Meerkat* [35], a computer vision approach to website defacement detection. The technique is capable of identifying

malicious content changes from screenshots of the website. Other studies focus on malicious redirectors and attack infrastructures. Examples include *JsRED* [77] that uses a differential analysis to automatically detect malicious redirect scripts, and *Shady Path* [111] that captures a malicious web page by looking at its redirection graph. Compared with those techniques, our approach is different in that it automatically analyzes the semantics of web content and looks for its inconsistency with the theme of the hosting website. We believe that the semantics-based approach is the most effective solution to promotional infections, which can be easily detected by checking the semantics of infected sites but hard to identify by just looking at the syntactic elements of the sites: e.g., both legitimate and malicious ads can appear on a website, using the same techniques like redirections, iframe, etc. Further, we do not look into web content or infrastructure at all, and instead, leverage the search results to detect infections. Our study shows that this treatment is sufficient for finding promotional infections and much more efficient than content and infrastructure-based approaches.

Similar to our work, *Evilseed* [66] also uses search results for malicious website detection. However, the approach is only based upon searching the URL patterns extracted from the malicious links and never touches the semantics of search results. Our study shows that focusing only on the syntactic features such as URL patterns is insufficient for accurate detection of promotional infections. Indeed, Evilseed reports a huge false detection rate, above 90%, and can only serve as a pre-filtering system. On the other hand, our technique inspects all the snippet of search results (not just URLs), automatically discovering and analyzing their semantics. This turns out to be much more effective when it comes to malicious promotional content: SEISE achieves low FDR (1.5%) at a detection coverage over 90%.

CHAPTER III

TOWARD AUTOMATIC GATHERING OF OPEN-SOURCE CYBER THREAT INTELLIGENCE

3.1 Introduction

According to Gartner, *Cyber Threat Intelligence* (CTI) is defined as "evidence-based knowledge, including context, mechanisms, indicators, implications and actionable advice, about an existing or emerging menace or hazard to assets that can be used to inform decisions regarding the subject's response to that menace or hazard" [99]. Such knowledge is essential for an organization to gain visibility into the fast-evolving threat landscape, timely identify early signs of an attack and the adversary's strategies, tactics and techniques, and effectively contain the attack with proper means. Given its importance, CTI has been aggressively collected and increasingly exchanged across organizations, often in the form of *Indicators of Compromise* (IOC) [91], which are forensic artifacts of an intrusion such as virus signatures, IPs/domains of botnets, MD5 hashes of attack files, etc. Once collected, these IOCs can be automatically transformed and fed into various defense mechanisms (e.g., intrusion detection systems) when they are formatted in accordance with a threat information sharing standard, such as OpenIOC [18], that enables characterization of sophisticated attacks like drive-by downloads. The challenge, however, comes from the effective gathering of such information, which entails significant burdens for timely analyzing a large amount of data.

Finding IOCs online: challenges. While IOCs can be extracted from traditional blacklists, like CleanMX [42] and PhishTank [12], the information delivered by such IOCs is rather thin: only a small number of IOC classes are covered (URL, domain,

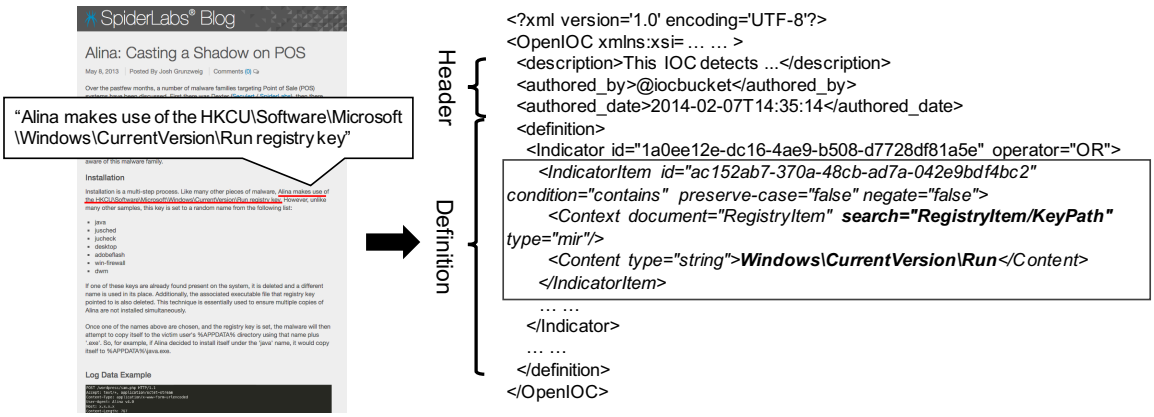


Figure 1: Example of OpenIOC schema.

IP and MD5), the relation between IOCs is not revealed and no context information is provided (e.g., the criminal group behind malfeasance). Analyzing the cyber-attack campaigns and triaging incident responses become quite difficult when relying on such information. Instead, IOCs from articles in technical blogs and posts in forums are more favorable to security practitioners and extensively harvested, since comprehensive descriptions of the attack are often found there. Such descriptions are typically informal, in natural languages, and need to be analyzed semantically to recover related attack indicators, before they can be converted into the standard IOC format such as OpenIOC, as illustrated in Figure 1. For years, this has been done manually by security analysts. Increasingly, however, the volume and velocity of the information generated from these sources become hard to manage by humans in a cost-effective way. As an example, in our research, we studied over 71,000 articles from 45 technical blogs extensively used by security professionals and found that the number of articles posted here has grown from merely a handful back 10 years ago to over 1,000 every month since last year (see Section 6.3.2). Note that these blogs are just a drop in the bucket: for example, Recorded Future is reported to utilize over 650,000 open web sources in 7 languages to harvest IOCs [97]. With the huge amount of information produced by those sources, new technologies are in dire need to *automate* the identification and extraction of valuable CTI involved.

Automatic collection of IOCs from natural-language texts is challenging. Simple approaches like finding IP, MD5 and other IOC-like strings in an article, as today's IOC providers (AlienVault, Recorded Future) do, does not work well in practice, which easily brings in false positives, mistaking non-IOCs for IOCs: e.g., as illustrated in Figure 2, although three zip files show up in attack-related articles, `MSMSv25-Patch1.zip` is clearly not an IOC while the other two are. Further, even for a confirmed IOC, we need to know its context, e.g., whether it is linked to drive-by download (`ok.zip` in the figure) or Phishing (`clickme.zip`), in order to convert it to the IOC format and help an organization determine its response. This can only be done by establishing a relation between the IOC token and other content in the article, such as the terms "downloads", "attachment" in the example.

Identifying semantic elements (called *Named Entity Recognition* or NER [87]) and extracting relations between them (called *Relation Extraction*, or RE [32]) have been extensively studied in the Natural Language Processing (NLP) community. However, existing NLP techniques *cannot* be directly applied for IOC and its context discovery. NER systems are known to be brittle, highly domain-specific — those designed for one domain hardly work well on the other domain [104]. So far, we are not aware of any NER techniques developed specifically for recognizing IOCs and a direct use of the state-of-the-art tools like Stanford NER [53] leads to low precision (around 70%) and recall (less than 50%) (see Section 3.3). Further, the current study on RE focuses on the relation between two known entities, which are typically nominals [47], whereas the relations between an IOC and the terms describing its context (which we call *context terms*) are way more complicated: e.g., "downloads" and `ok.zip` has a verb-noun relation. Also important, the accuracy and coverage offered by the existing RE tools are typically low. For example, the classic tree kernel approach [47] reports a precision between 50 and around 90% and a recall between 10 and 50%. As another example, a recent proposal for extracting events among genes and proteins

from biomedical literature [104] has a precision and recall around 50%. This level of performance cannot meet the demand for high-quality IOCs, which could go straight to a security system to offer immediate protection for an organization.

iACE. A key observation from the open intelligence sources producing high-quality IOCs is that technical articles and posts tend to describe IOCs in a simple and straightforward manner, using a fixed set of context terms (e.g., "download", "attachment", "PE", "Registry", etc.), which are related to the *iocterms* used in OpenIOC to label the types of IOCs [18]. Further, the grammatical connections between such terms and their corresponding IOCs are also quite stable: e.g., the verb "downloads" followed by the nouns "file" and `ok.zip` (the IOC) with a compound relation; "attachments" and `clickme.zip` also with the compound relation. This allows us to leverage such relations to identify an IOC and its context tokens, combining the NER and RE steps together. To this end, we developed a new approach, called *iACE* (IOC Automatic Extractor), in our research, which tailors NLP techniques to the unique features of IOC discovery. More specifically, after preprocessing articles (using *topic-term classification* to find those likely involving IOCs and converting figures to text), iACE utilizes a set of regular expressions (regex) and common context terms extracted from iocterms to locate the sentences within the articles that contain *putative IOC tokens*, such as IP, MD5-like string. Within each of such sentences, our approach attempts to establish a relation between the IOC token and the context term: it converts the sentence into a *Dependency Graph* (DG) to describe its grammatical structure and extracts the shortest paths linking each pair of a context token and the putative IOC token; over each path, a *graph mining* technique is applied to analyze the relation between the tokens, which cannot be handled by existing RE techniques, for the purpose of determining whether they indeed include an IOC and its context.

This simple approach leverages the known "anchors" (context and regex terms)

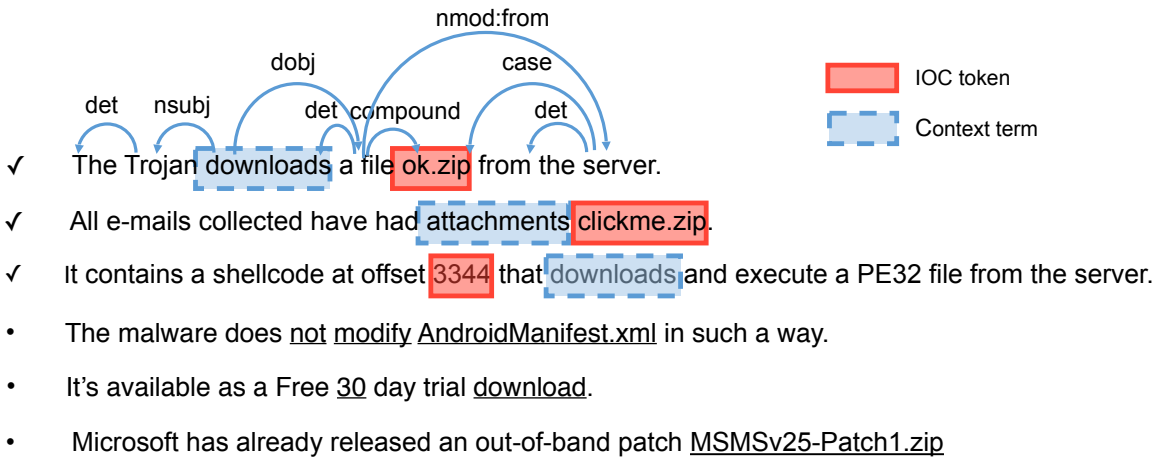


Figure 2: Examples of sentences with/without IOC.

to locate potential IOC-carrying sentences and the predictable relations among them to capture IOCs, avoiding the complexity of direct application of existing NLP techniques, which needs to solve the NER first to identify an IOC token before addressing the RE problem to find its context. Our evaluations on a prototype we built show that this new technique is very effective in practice: it achieved a precision of 95% at a coverage over 90%. In the meantime, our system is also highly efficient, capable of analyzing four articles per second even with a single process.

Our discoveries. Running on all 71,000 articles collected from the 45 blogs (including AlienVault, Malwarebytes, Webroot, etc.) in the past 13 years (from 2003/01 to 2016/04), iACE automatically extracted 900K IOC tokens together with their context. By inspecting these items and correlating them across different blogs over the long time frame, we were able to gain an unprecedented understanding of the relations between different attacks reported, the impact of open-source threat intelligence on attack evolutions, the defense responses it triggered, as well as the qualities of these blogs and effectiveness of the IOCs they document. More specifically, we found that some apparently unrelated attack instances were actually connected, sharing the same attack assets and infrastructures such as command and control (C&C) hosts. Particularly, by linking together 396 articles and more than 7,000 IOCs, our study reveals

that a C&C campaign continued to evolve over a four-year span, changing the targets of exploits from one vulnerability (CVE-2010-1885) to another (CVE-2013-0422). Also interestingly, we observed that the attackers might adjust their strategies in response to the release of IOCs: e.g., we found that the IOCs receiving intensive reports tend to be short-lived, typically disappearing after a month.

On the other hand, organizations do not seem to react quickly to the release of IOCs: it could take around 2 days for AV scanners to include the hash of new malware and over 12 days for web scanners to update their domain and IP blacklists, after related information was made public by the blogs. Also, we found that a buffer overflow vulnerability CVE-2012-0158 was first reported by AlienVault in April, 2012 for APT attacks on the military and aerospace industry and then showed up again in an article on TrendMicro, September 2012, for an attack on political organizations; later the same vulnerability was found in malware distribution (2013) and Spear Phishing campaigns on the film industry (2014) and banks (2015), indicating that such a long-standing IOC was not adopted timely.

In terms of the qualities of open-source intelligence, we found that *Hexacorn* and *Naked Security* often provide timely and comprehensive information about new attacks. For example, articles from Naked Security first reported three malicious name servers `m.sea.sy`, `mod.sea.sy` and `sea.sy` under the control of the Syrian Electronic Army for DNS hijacking. Also, some IOCs apparently are more predictive than others. An example is the name server "132.248.49.112", which was reported by 19 blogs for the multiple-theme Spear Phishing attack and remained unchanged for 140 days.

Contributions. The contributions of the work are as follows:

• *Novel IOC discovery technique.* We present iACE, the first fully-automated technique for generating OpenIOC compatible, semantic-rich intelligence, which addresses the emerging challenge in the effective analysis of massive open-source data for timely CTI gathering. Our approach leverages the unique features of IOCs and the way

they are described in mainstream technical articles to come up with a specialized, scalable information extraction technique that achieves high accuracy and coverage. Our evaluation shows that iACE can effectively recover valuable attack indicators from popular technical blogs and convert it into industry-standard, machine-readable threat intelligence, which cannot be done by any existing techniques, to the best of our knowledge.

• *New findings.* Running iACE on over 71,000 articles from 45 most popular technical blogs across 13 years, our study sheds new light on the effectiveness of such open-source intelligence exchange, and the impact that it may have on the security industry and the adversary's strategies. The new understandings are invaluable for improving the IOC release, discovery and utilization process, contributing to the better protection of organizations' information assets.

3.2 Design and Implementation

Although the generic problem of information extraction from natural language text is hard, particularly when high accuracy and coverage are required, articles documenting IOCs tend to utilize common technical terminologies (e.g., IP, process, etc.) and present the artifacts and their context in a predictable way (e.g., utilizing compound dependency, see the example in Figure 2), thereby making identification of IOCs more manageable. These unique features were fully leveraged in the design of iACE, which utilizes high-confidence context terms from iocterms and a set of regexes to locate the sentences likely containing IOCs. Then iACE analyzed the relations just between those anchors (the context terms and the putative IOC token) against a model learned off-line to accurately and efficiently capture IOCs. Below we present the high-level design of this technique and explicate how it works through an example.

Architecture. Figure 3 illustrates the architecture of iACE, including a blog scraper (BS), a blog preprocessor (BP), a relevant-content picker (RCP), a relation checker

(RC) and an IOC generator (IG). The BS first automatically collects ("scrapes") technical articles from different technical blogs, removing irrelevant information (e.g., advertisements) from individual blog pages. These articles are then inspected by the BP, using NLP techniques to filter out those unlikely to contain IOCs. For each article considered to be IOC-relevant, the RCP converts all its content, including pictures, to text when possible, breaks the content into sentences and other special content elements (tables and lists) and then searches among the sentences for those likely involving IOCs with a set of context terms and regexes, which describe the formats of IOCs such as IP addresses. The context terms here are automatically extracted from iocterms and can also come from other manually labeled sources, and the regex is built specifically for each IOC type and its consistency with the content terms is first checked before the sentence is selected. For each sentence, the RC analyzes its grammatical structure connecting the context anchors (context terms) and the IOC anchor (the string matched by the regex), using a learned model to determine whether the latter is indeed an IOC, and if so, mark the IOC and the context terms. Using such labeled content, the IG automatically creates the header and the definition components, including the indicator items for all the IOCs identified, according to the OpenIOC standard.

An example. Here, we use the excerpts of an article posted on Trustwave (Figure 1) to go through the whole IOC extraction procedure. From the title and other keywords identified from the article, using content term extraction (an NLP technique), iACE detects that the article is likely to contain IOCs for a malware infection, and then marks the link content and summarizes the article using *TextRank* [83] to construct the header component. After that, it converts the figures in the article into text using *Tesseract* [11], and searches for the context terms and IOC token on each sentence.

Among all the sentences captured, the following one is found to include a string

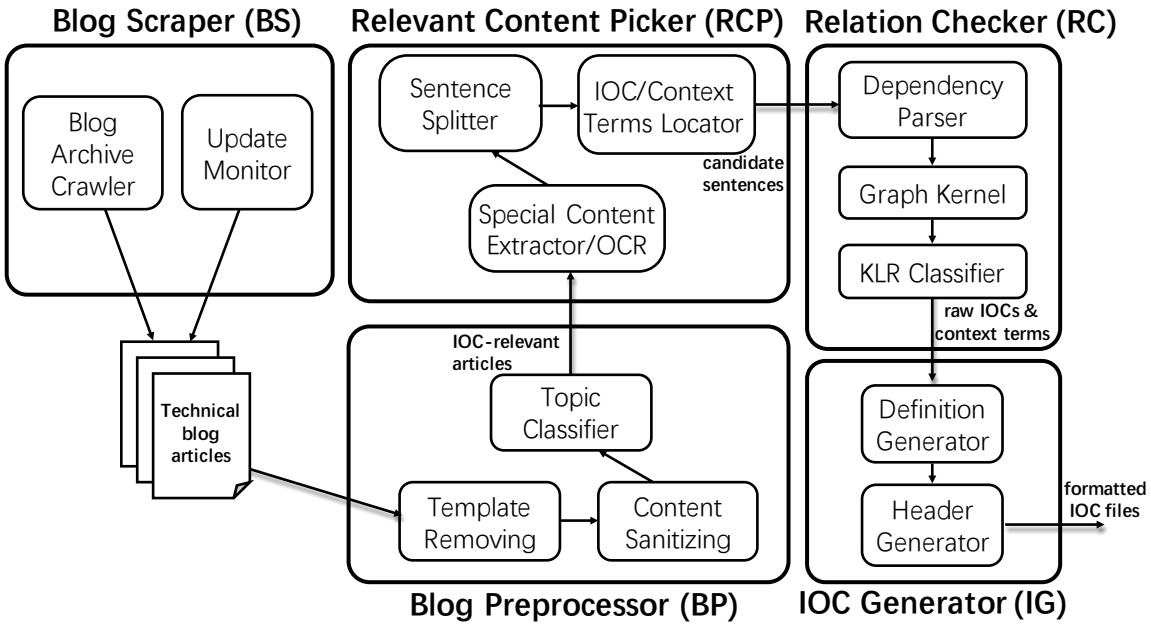


Figure 3: The architecture of iACE.

fitting the description of a key path of registry, along with the context terms "registry" and "path": "It includes the addition of the registry value on the path `HKCU\Software\Microsoft\Windows\CurrentVersion\Run` that virus use to maintain persistence." This sentence is then analyzed against the IOC recognition model built through relation analysis, which confirms that the connection across the terms and the IOC anchor here ("`HKCU\Software\Microsoft\Windows\CurrentVersion\Run`") is indeed commonly used to describe the key path of an injected registry item. As a result, these terms and the IOC are labeled. From such content, the technical details of the attack reported by the article are automatically extracted and presented in the OpenIOC format (Figure 1).

Assumptions. The current design of our system is for analyzing technical blogs, under the assumptions that the blog writers present their findings in a professional fashion. Such assumption is reasonable intuitively, as the blogs are written by security professionals and their purpose is to quickly share technical details to their

peers[1]. It was also confirmed by our analysis of the labeled dataset `DS-Labeled` . All their descriptions of IOCs were found to be in line with our assumptions, involving professional terms and predictable grammatical structures. On the other hand, the effectiveness of the technique on other types of IOC open sources like online forums needs to be further studied.

3.2.1 Relevant Content Identification

As mentioned earlier, to automatically extract IOCs from technical blogs, we first need to scrape related web pages from blog sites, pre-process their content and remove noise, before filtering out those irrelevant and identifying from the remaining articles the sentences that may carry IOCs for the follow-up relationship check and IOC extraction. Here we elaborate on these individual steps.

Blog scraping and pre-processing. Our blog scraper is essentially a crawler designed to continuously monitor a list of technical blogs to collect their articles. For each blog site, the BS first scraps all its existing articles before it is set to monitoring mode to look for new ones. Specifically, the scraper performs breadth-first crawling on the blog, starting from its homepage to explore each link it discovers, until no new link can be found. A problem here is that the web pages gathered in this way may not all be articles, and may also contain login pages, contact pages, and others. To automatically remove these unrelated pages, we leverage two unique observations: the articles posted on the blog are all framed within the same HTML template, which is very different from those used by other pages such as login, contact, and others; also, on any blog site, more article pages are hosted than other types of pages. Based on the observations, the BS compares each page's DOM tree with those of others

[1]Note that asking those professionals to directly post formalized IOCs may not be realistic in the near future, due to the complexity of manually creating the content and their intent to get humans involved in the discussion. As a supporting evidence, we examined all 45 blog sites and found only one of them export IOCs in some articles (AlienVault).

to find out a small set of pages framed over the templates unlike the ones used by the majority of the pages. These pages are considered to be outliers and dropped from the dataset scraped from the blog site. In our implementation, the BS uses the python library `beautifulsoup` [2] to extract from each page its HTML template (with tags and attributes but no content) and groups the pages using their templates' hash values to identify those unrelated to articles (the ones outside the group).

Even on those indeed relevant web pages, there still is content unrelated to the technical articles, such as blog contributors' pictures, advertisements, featured content, etc. Such content needs to be removed before the pages can be used for IOC identification. This purpose is served by another pre-processing step the BS takes to get rid of such non-UGC (*user-generated-content*) data from each page. Specifically, our approach compares all pages' DOM trees to find out the nodes with the non-UCG, characterized by their largely unchanged content across these pages (e.g., blog contributors' photos will be the same across different articles). This is different from the article content generated by the user, which varies significantly between pages. Such a difference is captured by an information-theoretic metric called *composite importance* (CI) [125] that is used by the BS to measure the uniqueness of each node on each page's DOM tree with regards to other pages: the nodes found to be less unique (rather common across different pages) are discovered and dropped from the DOM tree, using an algorithm proposed in the prior research [125].

Under such a "sanitized" DOM tree, still some content cannot be directly analyzed by our text-based approach, including images and embedded PDF files. So, the last pre-processing step is to convert such content into text, if it indeed involves text information. To this end, we incorporated into our implementation of an optical character recognition engine *Tesseract*. Tesseract [11] is capable of discovering texts within an image with an accuracy of 99% in general. However, for the images collected from blogs, we found that the accuracy went down to merely 70%, due to the low

quality of the images and the non-dictionary words they often have. To address this issue, we used *Gimp* [8] to resize the blog image a for better image quality, and add to Tesseract the new words (e.g., IP, MD5, HTTP) discovered from the text of an article. We tested this approach on 100 randomly sampled images from 10 blogs, which was found to push the accuracy of Tesseract to above 95%.

Topic filtering. Once all the article pages have been selected and pre-processed, we start looking into their content, first removing those not including any IOCs. Examples of such articles are those for product promotion, news or software update, which we call *non-IOC articles*. To separate the non-IOC articles from those with IOCs (called *IOC articles*), iACE runs the TC, a classifier using a set of features as described below:

• *Topic words*: Intuitively, an IOC article focuses on security risks and vulnerabilities, whose theme is reflected by its *topic terms* (e.g., malware, exploit). These terms are less likely to appear in a non-IOC article. *Topic term extraction* is an extensively studied NLP technique. In our implementation, we utilized an open-source tool `topia.termextract` [19] to collect the terms within each article. The top 20 terms discovered, together with their frequencies, are part of the features for the classification.

• *Article length*: Since the blog sites are meant to be the channels for IOC exchanges, the IOC article it contains tends to be longer, including detailed descriptions of IOCs and their context, while non-IOC articles in technical blogs are often news and digests, and hence tend to be shorter.

• *Dictionary-word density*: Compared with non-IOC articles, IOC articles tend to have a lower dictionary-word density, because most IOCs are non-dictionary words (e.g., IP, hash values, file path). Our implementation employs the `enchant` library [28] to find dictionary words in an article. Then, its density within the article is calculated as the ratio of the dictionary words with regards to all the words the article contains.

Using these features, we ran a support vector machine (SVM) to train a classification model over `DS-Labeled`, including 150 IOC articles and 300 non-IOC articles. The model was evaluated through a 10-fold cross validation, and found to achieve a precision of 98% and a recall of approximate 100%. We further used this TC to analyze all 71,000 articles from the unknown set `DS-Unknown` (described in Section 6.3.2) and manually validated 500 instances randomly selected from the classified results (Section 3.3.2). The validation shows that the classifier had a high accuracy (96%) and coverage (99%).

IOC sentence identification. From each IOC article, the relevant content picker identifies the sentences, tables, and lists likely to include IOCs before they are further evaluated by the relation checker (Section 3.2.2). Such content is selected from the article based on the *anchors* they contain, i.e., context terms and putative IOC tokens. Specifically, the RCP parses the HTML content of each article, breaking the text into sentences and detecting tables and lists. From each sentence, we look for the presence of both the string matched by the regex and its compatible context terms: e.g., "hash" goes with an MD5 checksum. From tables and lists, we increase the search scope for context terms also to table headers, captions and the nearest sentences (see supplementary material). Figure 2 shows an example.

The OpenIOC standard specifies 600 categories of IOCs using a list of iocterms [18]. We further summarized these IOCs into 19 different data types, including `IP`, `hash`, `int`, `float`, and others. The regex for each of these categories was manually constructed and carefully examined. As we can see here, such expressions could introduce false positives (e.g., the string type matching many IOC strings, even though oftentimes, we only seek non-dictionary words), if we do not also look at the context terms and the relations between identified tokens. These terms are automatically collected from the 600 iocterms through tokenizing dictionary-word elements within each iocterm and by removing common terms like "item" . We further used *Semanticlink* [27]

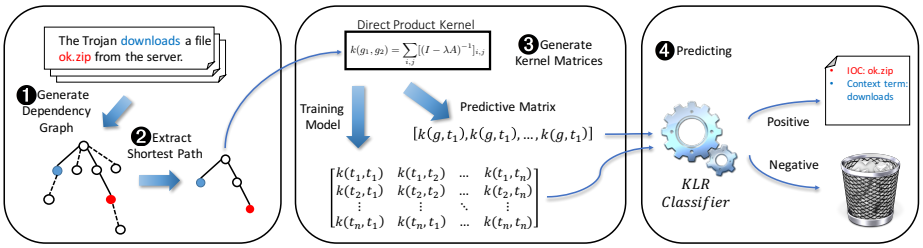


Figure 4: Workflow of the relation checker (RC).

Table 2: Examples of regexes.

General type	Regex
IPv4	`(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)(/([0-2][0-9]|3[0-2]|[0-9]))?`
hash	`\b[a-fA-F\d]{32}\b|\b[a-fA-F\d]{40}\b|\b[a-fA-F\d]{64}\b`
int number	`((?<=[\s([\s\(\[\{:\+\-=\\])|^)(\+|-)?\d+(?!([a-zA-Z0-9]|\.\S))`
float number	`((?<=[\s\(\[\{:\+\-=\\])|^)(\+|-)?(\d+)?\.(\d+)(?!([a-zA-Z0-9]|\.\S))`

to recover other semantic related terms, e.g., "portable executable" (related to PE).

Altogether, we gathered 5,283 context terms in our research, which were found to indeed provide good coverage of the common terminologies used by technical blogs. In our research, we gathered 80 public IOC files and their corresponding blog articles (according to their description tags) from labeled dataset. By manually inspecting the sentences carrying the IOCs, we found that all such sentences also contain at least one context term and all such terms are on the list we created.

3.2.2 Relation Checking and IOC Creation

Although context terms and regexes can find us the sentences likely involving IOCs, they are *insufficient* for detecting true IOCs with high accuracy. For example, Figure 2 shows four sentence pairs collected from two articles posted on AlienVault. As we can see, each pair contains both context terms, like "download", and the strings fitting the descriptions of their corresponding IOCs, such as "3344"; however, the fifth one does not include any real IOC while the third one does. Fundamentally, the coincidence of relevant tokens (context term and IOC token) does not necessarily

indicate the presence of an IOC-related description. Most important here is the consistency between the *relation* of these tokens and what is expected from the narrative of IOCs: in the above example, `3344` is supposed to be the *offset*, not the *PE* file. Actually, in the case that such a relation is incorrect, even when the IOC extracted is indeed an attack indicator, its context can be wrong and as a result, the OpenIOC record created can be erroneous. The third sentence in the figure is such an example.

As mentioned earlier, identifying IOCs is essentially an NER problem and connecting them to their context is an RE issue. In the NLP community, solutions to the problems are pipelined: individual name entities within a sentence are first recognized (i.e., NER) and then their relation is established (i.e., RE). For our problem, however, this pipeline becomes unnecessary, since the putative tokens for an IOC and its context are already located in a sentence, and all we need to do is check the consistency of their relation with what is expected to happen between them. In other words, we are in the position to address both the NER and RE together. On the other hand, existing RE techniques are inapplicable here, because they are designed to work on the nominal relation between two nouns, whereas the links between an IOC and its context terms are more diverse, including nominal, verb and adjective (e.g., "attachment", "download" and "injected"). To handle such diverse relations, we came up with an idea that models the analysis on the grammatical connection between the IOC candidate and its context as a *graph mining* problem [45]. This allows us to apply graph similarity comparisons to detect the presence of the desired relation, which achieves both a high accuracy (95%) and a high coverage (above 90%) that the more generic RE techniques cannot attain [47]. Below we present how the approach works.

Relation representation. To analyze the relation between an IOC candidate and a context term, our approach first uses a *dependency parser* to transform a sentence into a DG. A DG describes the grammatical connections between different linguistic

tokens, with verbs being the structural center of a clause and other tokens directly or indirectly depending on the center. Such a dependency describes the grammatical connection among tokens such as direct object, determinant, noun compound modifier, etc. Formally, a DG is a directed and weighted graph $g = (V, E, W)$, where words in the sentence are nodes V, two related nodes are connected by an edge E and the specific grammatical relation linking them together is modeled as an edge weight W. In our research, the DG was constructed using the Stanford dependency parser [39], the state-of-the-art open-source tool that converts text to simplified grammatical patterns suitable for relations extraction.

Unlike the more generic RE techniques, which work on the DG of the whole sentence, our approach takes advantage of the known anchors to focus on the smallest dependency graph g_{n_i,n_c} connecting an IOC candidate n_i and a context term n_c together. This is because the information carried by this subgraph (called *core*) is most relevant to the understanding of the relations between the anchors, which is all we care about (see an example in Figure 4). Note that we also add negation dependencies as child nodes or sibling nodes of the nodes on the core to capture IOC-related negative descriptions. As an example, in the fourth sentence in Figure 2, the relation between the context term "modify" and the IOC token "`AndroidManifest.xml`" is affected by the word "not", which adds a negation dependency on the verb "modify".

Similarity comparison. Over a dependency subgraph g_{n_i,n_c} modeling the relation between a context term and an IOC candidate, we want to find out whether another subgraph is similar, which indicates that the same relation also exists between its anchors. This similarity comparison is important, since it is the foundation for classifying a sentence, determining whether IOC relations are indeed there to bind anchors or they are not. Given the fact that now we only need to work on simple subgraphs with a few labeled nodes (see Figure 4), the focus is to compare the paths linking corresponding nodes (with identical labels) across the subgraphs. For this purpose,

we customize a well-known graph mining technique, called *direct product kernel*, a function that measures the similarity of two graphs by counting the number of all possible pairs of arbitrarily long random walks with identical label sequences [54].

Specifically, let $g_1 = (V_1, E_1, W_1)$ and $g_2 = (V_2, E_2, W_2)$ be directed weighted graphs. The direct product between g_1 and g_2 is a directed weighted graph $G = g_1 \times g_2 = (V, E, W)$, where

$$V = \{(v_1, v_2) \in V_1 \times V_2\}$$

$$E = \{(u_1, u_2), (v_1, v_2) \in V \times V : (u_1, v_1) \in E_1, (u_2, v_2) \in E_2\}$$

$$W = \begin{cases} 1, & \text{if } W_1(u_1, v_1) = W_2(u_2, v_2) \\ 0, & otherwise \end{cases}$$

In other words, for each pair of nodes (u_1, u_2) and (v_1, v_2) in the new graph G, they are adjacent (connected by a directed edge) if and only if an edge with the *same* direction and weight exists both between u_1 and v_1 in g_1 and between u_2 and v_2 in g_2. It is important to note that we bring the weights into the product to compare the *type* of the grammatical relation between two nodes (words): only when the adjacent word pairs in two sentences have the same grammatical relation, will that relation be preserved in the new graph. Usually, it takes $O(|V|^4)$ to compute this direct product. However, all the subgraphs we consider are just "paths" (after removing the directions) between a context term and an IOC candidate, whose direct product can be computed in $O(|V|^2)$.

Over the direct-product graph G, we calculate the similarity between the two subgraph g_1 and g_2 as follows:

$$k(g_1, g_2) = \sum_{i,j} [\sum_{l=0}^{\infty} \lambda^l A^l]_{i,j} = \sum_{i,j} [(I - \lambda A)^{-1}]_{i,j}$$

where each entry of A^l is the number of the walks of length l from v_i to v_j in the direct product graph $G = g_1 \times g_2$, hence A^l can be calculated as the l-th power of the

adjacency matrix of G; λ is the decay constant and $\lambda \leq \frac{1}{min(d_i,d_o)}$ with d_i, d_o being the maximum in-degree and out-degree of G, representatively. In our research, we set $\lambda = \frac{1}{min(d_i,d_o)}$.

Classification. Based on our customized direct-product kernel, we run a classifier (the relation checker) to determine the presence of IOC relations between a context term and an IOC candidate within a sentence. The classifier is trained over `DS-Labeled` with 1500 positive instances and 3000 negative instances. From the dataset, a model is learned to work on a kernel vector generated from the features of a subgraph g_i: $(m_1, \cdots, m_j, \cdots)$, where $m_j = k(g_i, t_j)$ and t_j is the subgraph for the jth instance in the training set. In other words, each new instance g_i is classified into the positive set (with the IOC relation) or the negative set (without the relation) based on its similarity with every instance in the labeled set. This classification is executed efficiently with multi-threading in our implementation.

The classifier can be trained with different kinds of machine learning algorithms. Our implementation utilizes logistic regression, since the algorithm works well on a small labeled dataset. More specifically, it optimizes the log likelihood function to determine a probability that is a logistic function of a linear combination of the training dataset, and every training point has a certain influence on the estimated logistic regression function. In our research, we compared the recall and precision of five classification models on the labeled dataset through a 5-fold cross-validation. With the regularization parameter set to 3.0, our logistic regression classifier yielded the best results. On unknown set `DS-Unknown` , this classification model achieved a high accuracy (with a precision of 95% and recall of 90%).

IOC creation. After identifying the IOC and its corresponding context terms, our IOC generator can automatically convert the CTI content of a technical blog to the OpenIOC record. Specifically, each indicator item in the record is created by filling in the `search` attribute in the `Context` tag with a context term and the `Content` tag with

its corresponding IOC. The content of other fields on the item can be derived from these two fields. For example, the `type` attribute of the `Content` tag and the `document` attribute of the `Context` tag are actually the iocterms of the IOC discovered.

For the header of the record, the IG generates the content for the `Description` tag using the open-source automatic summarization tool `TextRank` to create a summary for the blog article. Also, the original blog link address is used to fill the `link` tag, and the content of `authored_by` and `authored_date` tag are set to our prototype name and the file generation time.

3.3 Evaluation

3.3.1 Settings

In our study, we ran our implementation of iACE to automatically analyze 71,000 real-world technical articles, on an R730xd server with 40 of Intel Xeon E5-2650 v3 2.3GHz, 25M Cache CPUs and 16 of 16GB memories. Here we explain the datasets used in the study and the parameter settings of the system.

Datasets. We utilized two datasets in our study: a labeled set for training our topic classifier (Section 3.2.1) and relation checker (Section 3.2.2), and an unknown set for evaluating our technique.

• *Labeled dataset (`DS-Labeled`).* The dataset contains 80 IOC files and their corresponding blog articles. These IOC files were collected in our research from two public feeds, iocbucket [67] and openiocdb [1], both providing threat intelligence through OpenIOC items. Under the `description` tags of these items, we found the links pointing to these items' sources and recovered 150 articles from 22 blogs, including their HTML pages and image files. Also, we manually gathered, from the same blogs, 300 other articles. Each of them was manually checked to ensure that they do *not* have any IOCs but contain some IOC-like strings (e.g., IP addresses, MD5, etc.). Most of them are technical news or articles for product promotion.

From these articles, we further extracted two kinds of sentences, those with IOCs (*true IOC sentences*) and those without but involving IOC-like strings (*false IOC sentences*). More specifically, the 1,500 true IOC sentences were identified using the context terms and IOC tokens specified in the OpenIOC items we collected, and further manually inspected to ensure their correctness. The 3,000 false IOC sentences were those matched by the regular expressions and context terms used by iACE (Section 3.2.1) but did not include any IOCs, as confirmed by a manual check.

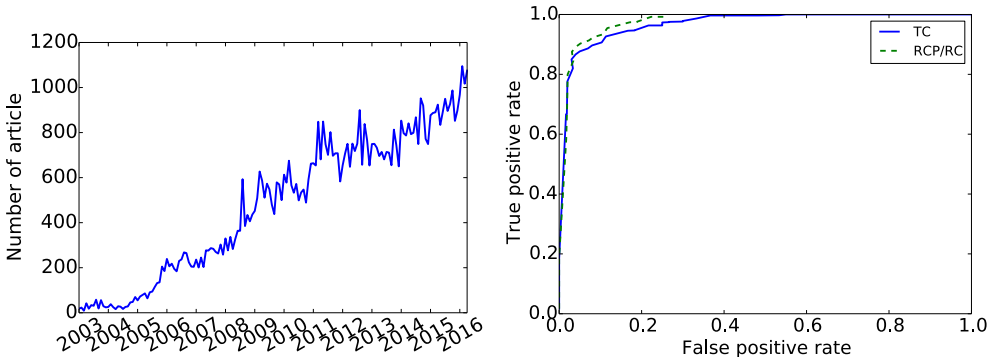
• *Unknown set (`DS-Unknown`).* As mentioned earlier, the dataset used for evaluating our system was gathered from 45 security-related technical blogs. These blogs are well recognized to be the leading sources for CTI collection, which includes AlienVault, Malwarebytes, and others (see supplementary material for the full list). On these blogs, we ran a crawler that scraped 71K articles posted there between 2003/04 and 2016/05. Figure 5(a) shows the increase of the number of the articles per month on these blogs over 13 years.

Parameter settings. In the experiments, the parameters of our prototype system were set as follow:

• *Kernel decay constant (λ).* The decay constant is a parameter for calculating a direct product when the impact of a long random walk needs to be discounted (Section 3.2.2). It was set according to the convention of evaluating the kernel function: $\lambda = 0.9$ if $\frac{1}{min(d_i,d_o)} = 1$ and $\lambda = \frac{1}{min(d_i,d_o)}$ otherwise.

• *Inverse of regularization strength (C).* Regularization is a parameter for reducing the over-fitting to the labeled data when we built the relation models using logistical regression. In our implementation, we utilized a $C = 3.0$, which gave the best performance among other C values from 0.3 to 15.

• *Threshold of sentence length (l).* The current dependency parser is limited by its capability to process long sentences. When a sentence grows longer, the accuracy of



(a) **The number of the articles per month.** (b) **ROC curve of iACE.**

Figure 5: The increase in the number of articles over 13 years and the effectiveness of iACE.

parsing goes down and its overhead goes up. In our implementation, we set the maximum length of the sentence to 200 words. Given IOCs may exist in sentence length larger than 200 words, we directly extract IOCs from sentences longer than 200 words without building dependency graphs. In our implementation, we directly extract the IOCs from sentences longer than 200 words if the sentence has two continuous IOC tokens or has more than five IOC tokens which were split by short terms (less than 5 characters) (see supplementary material).

3.3.2 Results

Accuracy and coverage. In our study, we first evaluated the topic classifier (i.e., TC) over the 450 IOC and non-IOC articles, and the relevant content picker and the relation checker (i.e., RCP/RC) over 1,500 true IOC sentences and 3,000 false IOC sentences in `DS-Labeled`, both using a five-fold cross-validation. Our prototype achieved a precision of 98% and a recall of 100% in finding IOC articles, and a precision of 98% and a recall of 92% in identifying true IOCs and its context. Figure 5(b) illustrates the ROC curve of the RCP and RC.

Further, we ran our system over `DS-Unknown` across all 71K articles. Altogether, iACE automatically extracted IOC tokens and their context terms, and further converted them into OpenIOC items. To understand the accuracy and coverage of the

information extracted, we sampled the unknown set using two different methods: we first grouped the articles in the unknown set according to their publication time (4 consecutive months per group), and then their publishers (45 blogs), and in each case, we randomly picked up a few articles from each group. Altogether, in this validation step, we manually inspected 820 articles and in total, 25K reported sentences, and concluded that iACE achieved a precision of 95% (23K out of 25K reported IOCs were correct), and a recall of 90% (across the articles, 90% of IOCs were reported).

Table 3: The number of samples and average accuracy and recall in each method.

Method	# of Groups	Samples per Group	Precision	Recall
Time span	39	10	93%	92%
Blog	45	10	96%	91%

To compare our approach with state-of-the-art alternatives, we ran iACE against the top-of-the-line NER tool Stanford NER [53] and the commercial IOC identifier integrated within AlienVault OTX [30]. Note that none of them (actually none of the existing systems we are aware of) can also identify the context for an IOC and therefore generate machine-readable OpenIOC items. In our experiment, Stanford NER was trained on our labeled *true/false IOC sentences* as describe in Section 6.3.2 (the same set for training iACE). For AlienVault OTX, we utilized its API to submit articles to their web service and retrieve the IOC tokens identified. This study shows that our relation-based approach is indeed much more effective. Specifically, we ran all three systems on 500 randomly selected articles from 25 blogs in `DS-Unknown` and compared their findings: iACE extracted the IOC items across 427 OpenIOC categories, such as `FileDownload HistoryItem/FileName`, `Email/ReceivedFromIP`, with a precision of 98% and a recall of 93%, while OTX could only find the IOCs in 8 categories (IP, hash value, domain, etc.) with a precision of 72% and a recall of 56%, a performance mirrored by Stanford NER. Both OTX and Stanford NER tend to introduce a lot of false positives: for example, OTX treated the reference links of articles as malicious URLs. Table 4 summarizes the result of this study.

Table 4: Accuracy and coverage comparison of iACE, AlienVault OTX and self-trained Stanford NER.

Tool	Precision	Recall
iACE	98%	93%
AlienVault OTX	72%	56%
Stanford NER	71%	47%

Performance. To understand the performance of iACE, we measured the time it spent on each real-world article in the unknown set and the breakdowns of the overhead in each analysis stage, BP, RCP, RC, and IG. In the experiment, our prototype was running on our R730xd server, using 40 threads. On average, it took around 0.25 second to inspect one article, as illustrated in Table 5. This result provides strong evidence that iACE can easily scale to the level expected for processing a massive amount of CTI generated every day.

Table 5: Running time at different stages.

Stage	average time (ms/article)	Std Deviation (ms)
BP	5	-
RCP	20	2.5
RC	252.5	37.5
IG	2.5	-
total	278.6	-

In the meantime, considering the DG parser is largely affected by sentence length, we measure the performance of the RC model in different sentence lengths. Figure 6 illustrates the average running time on sentences of different lengths. We found that the running time of iACE gradually increases when sentence length increases. This is because iACE only extracts the shortest paths linking each pair of a context token and the IOC token, which mitigates the impact of sentence length .

3.4 Measurement and Analysis

The IOCs automatically extracted from the 71,000 articles on 45 blogs present to us a comprehensive view of the cyber threats that the world has been facing in the past 13 years. By analyzing these IOCs, looking at their relations and evolution over a large time frame, across thousands of articles, our study brings to light new findings of

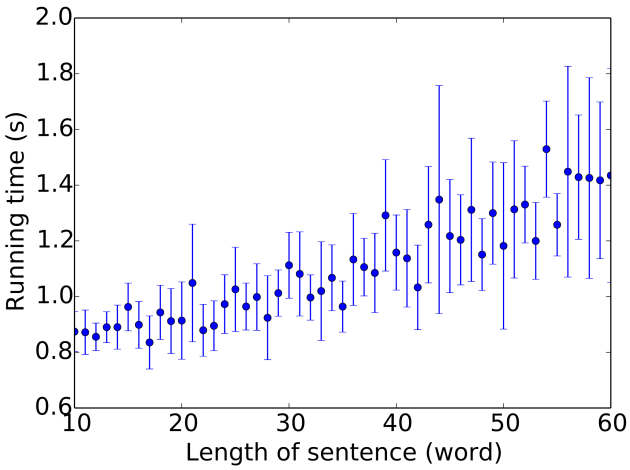


Figure 6: Average running time on the sentences in different lengths.

attacks' strategies and evolution, as well as new insights into the impact of open-source intelligence. Particularly, we found that hundreds of apparently independent attacks were actually related, sharing unique IOCs such as IP address, register's email and domain, etc. Among them, a set of command and control (C&C) servers were reported by 396 articles (with little reference among them) and linked to over 7,000 unique IOCs over a four-year span, indicating that these separate attacks could all be part of a massive, previously unknown campaign. Further, through correlating different articles, we observe that the same vulnerability has been continuously utilized for a long period of time. Also, the IOCs intensively reported tend to be short-lived, demonstrating the possible impacts of the CTI on the adversaries. On the defender side, the response to the IOCs seems less timely than one hopes, taking days to get IOCs into malware scanners, blacklists etc. Also, our study reveals the quality of the IOCs reported by different blogs and the ways these blogs react to emerging threats, which helps better understand the effectiveness of such intelligence sources. Below we elaborate on the details of this study.

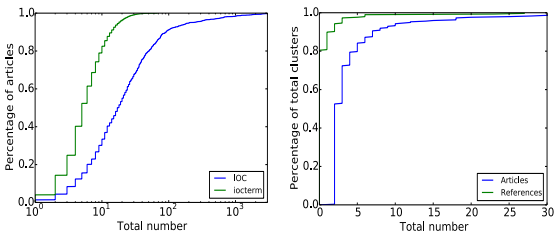
3.4.1 Landscape

Our study shows that these technical blogs are indeed a gold mine for threat intelligence gathering: altogether, 900K IOCs and their context were recovered from 71,000 articles (20K IOC articles), including 20K exploit hash values, 55K registry key, 58K malicious IPs, 180K FQDNs etc. Table 6 presents the 10 IOC types (described by their corresponding iocterms) with the most instances found by iACE. We observe the largest amount of IOCs with the type PortItem/remoteIP, which was related to the popularity of drive-by-download, Phishing and other web attacks in the wild.

Table 6: Top-10 iocterms with the largest number of IOCs.

iocterm	# of IOCs	general type
PortItem/remoteIP	18,243	IP
RegistryItem/ValueName	12,572	string
UrlHistoryItem/HostName	12,020	URL
RegistryItem/Path	11, 777	string
FileDownloadHistoryItem/SourceURL	10,324	URL
ServiceItem/serviceDLLmd5sum	9,908	hash
PortItem/remotePort	9,831	int
FileDownloadHistoryItem/FileName	8,720	string
Email/ReceivedFromIP	8,225	IP
FileItem/FileExtension	8,100	string

We looked into the distribution of IOCs across different articles and blogs, as illustrated by the cumulative distributions for the numbers of IOCs and iocterms displayed in Figure 7(a). On average, each article contains 52 IOCs and 70% of the articles have more than 10 IOCs. Particularly, the blog *hpHost* has 350 IOCs per article, the largest one among the blogs we inspected. Also, an article on *CyberCrime* talking about a Google redirection Spam reported 3,417 IOCs. When it comes to the diversity of IOC context, we found that on average, each article includes 6 different iocterms and 30% of articles have more than 10 different iocterms. Also interestingly, the blog with more IOCs does not necessarily come with more diverse IOC types: for example, even though *hpHost* has the largest number of IOC per article, each article only has 8 iocterms.



(a) **Cumulative distributions for the numbers of IOCs and iocterms per article.** (b) **Cumulative distribution of the numbers of articles and reference per cluster.**

Figure 7: Distribution of IOCs across different articles and clusters across different blogs.

3.4.2 Understanding Threats

Through mining the IOCs across articles and blogs, we gained new insights into reported attacks and discovered connections never known before. Such connections shed new light on the way the adversaries organize their campaigns and adapt their techniques. Further linking the reported IOCs to auxiliary data sources reveals the impact of such CTI on the responses to emerging threats.

Correlation analysis. To understand the relations across different attack campaigns reported by articles, we studied the sharing of critical attack resources in these campaigns, through measuring their common infrastructure-related IOCs, including IP, register's email and domain. Specifically, using these IOCs, we were able to group all the articles into 527 clusters (each group with more than three articles): two articles were put in the same cluster if they share at least one IOC IP, email or domain. Note that we removed IP addresses in the private address ranges (10.*.*.*, 172.16.*.*, 192.168.*.*). To find out whether those in the same cluster actually refer to a common source, so they essentially talk about the same attack instance, we also looked at the URLs in each of the articles to identify those pointing to the 45 blogs and other well-known intelligent sources. Figure 7(b) illustrates the distribution of the clusters with various percentages of the articles involving such references. It turns out that

many clusters (including thousands of articles) have low reference percentages: in other words, the authors of these articles apparently did not realize that the attacks they were documenting were related to other instances.

We further picked out 5 clusters with at least 25 articles but at most 5% of them include references to others. None of such reference-carrying articles were found to mention the relations among the attack instances reported by other articles in the same clusters. Also, a sampling of these articles did not show any references to other blogs not on our list. This indicates that the infrastructure relations linking all these attacks together have never been reported before, which was confirmed by our additional search for related literature across the Internet. Table 7 provides the information about those clusters. Most interestingly, we found that for the IOC "132.248.49.112", a shared IP reported by 19 blogs, turned out to point to a C&C campaign's name server. We believe that all the independently documented attacks actually belonged to a massive campaign whose scale was not known before. Note that this correlation effort is highly important because it informs us of the critical resources attackers share, which are likely to be their weakest link.

Table 7: The 5 Clusters.

Cluster	# of articles	# of IOCs of mal. infra.	# of IOCs of total	# of references
1	396	7,363	10,533	21
2	178	4,271	8,110	3
3	30	215	960	0
4	28	897	1,302	0
5	25	897	1,222	0

Evolution. Looking into the C&C campaign reported by 396 articles and related to 7,000+ unique IOCs, we were surprised to find that it lasted for a long time (2009-2013), and continued to adapt its attack strategies and techniques. Specifically, it distributed malware by sending Spam emails to users, compromising legitimate websites, and others, and the vulnerabilities it exploited evolved from CVE-2010-1885

to CVE-2013-0422. In the meantime, the campaign utilized a small set of IPs for its C&C servers, and some of them share the same register email "`gkook@checkjemail.nl`". Apparently, taking down these servers could significantly affect the effectiveness of this long lasting, large-scale attack.

Another step we took to correlate the attacks reported by different articles was to cluster them according to the indicators of their attack vectors, including malware hash, vulnerability CVE and the content of the registries. The purpose is to understand the evolution of the vectors with regard to the releases of related IOCs. To this end, we looked at the shortest period of time, in terms of the number of consecutive months, during which a specific IOC (e.g., 132.248.49.112) was continuously covered by new articles. This period, which we call "decay time", demonstrates how long the attack instances related to the IOC continue to pop up before they can be stopped (at least temporarily). As illustrated in Table 8, in general, we found that the IOCs reported by a large number of articles tend to disappear quickly, indicating that either the related problems were quickly fixed or the adversaries reacted to the reports to change their strategies. However, there are long-lasting IOCs even though they are well known: as a prominent example, a buffer overflow vulnerability CVE-2012-0158 kept showing up in blog articles for four months, and after a few recesses, continues to appear in other attacks over a four year period! Specifically, it was used in APT attacks on the military and aerospace industry (2012/04), and then on political organizations (2012/09), generic malware distribution (2013), and more recently Spear Phishing attacks on the film industry (2014) and banks (2015). This clearly shows that organizations have not done their due diligence to adequately respond to the problem.

IOC impacts. Further, we studied the impact of such open-source intelligence on security protection: whether the security industry (e.g., anti-virus service providers) quickly responded to the reports of IOCs. To this end, we estimated the possible

Table 8: Decay time of IOCs.

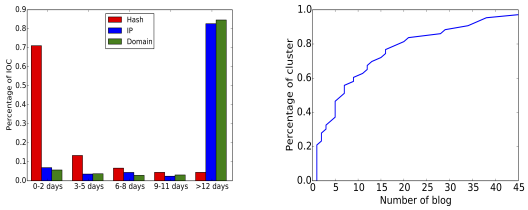
Decay time (month)	Avg. # of articles per month	% of total IOCs.
0-1	68	92%
1-2	23	5%
2-3	8	1%
3-4	6	1%
¿4	3	1%

time intervals between the first release of the IOCs and their adoption by anti-virus (AV) tools and web scanners. In our research, we submitted a set of IOCs, including IPs and domains of malicious sources, and the hashes of malware to *VirusTotal* [20], a platform that hosts 56 mainstream AV scanners and CleanMX [42], an IP/URL scanner. From VirusTotal, we collected the time stamps for the first time when the submitted IOCs were seen by at least one of the AV systems. For CleanMX, we fetched its IP/URL blacklist database archived from 2009/01 to 2015/04. We found that 47% of the IOCs were updated to these systems before they were reported by the blogs. For the rest of them, Figure 8(a) shows the distributions of the duration after such an IOC was released and before it was first used for a scan. We observed days of delay before the IOCs were put in place for protection, if indeed the uploading time (or the time when the IOCs were first used for a scan) was close to the moment when the IOCs (IP, domains, hashes) were added to the systems. Particularly, for IPs and domains, the whole process often took more than 12 days. On the other hand, the malware hashes were often quickly added, in most cases within 2 days.

3.4.3 Understanding Intelligence Sources

The availability of the longitudinal data (the IOCs collected over a span of 13 years) also enables us to investigate the qualities of the indicators produced by different sources and their timeliness against new threats, as reported below.

Timeliness. Using the aforementioned attack clusters (see Table 7), we analyzed the distribution of the articles first reporting the attacks over different blogs, as shown



(a) **Distributions of the duration after an IOC was released and before it was uploaded for a scan.**

(b) **Cumulative distributions for the numbers of first-reported blog per cluster.**

Figure 8: IOC impacts and timeliness of blog.

Table 9: Quality of selected intelligence sources (10 out of 45)

Blog	% of covered IOCs	% of covered ioc-terms	% of timely IOCs	% of robust IOCs
Dancho Danchev	42%	62%	14%	84%
Naked Security	43%	55%	54%	45%
THN	38%	38%	41%	51%
Webroot	54%	**79%**	13%	84%
ThreatPost	26%	37%	52%	29%
TaoSecurity	**57%**	61%	31%	68%
Sucuri	34%	35%	43%	52%
PaloAlto	39%	44%	15%	**87%**
Malwarebytes	32%	48%	26%	72%
Hexacorn	49%	57%	**59%**	76%

in Figure 8(b). We found that 10 blogs were responsible for the first report of 60% the clusters (each cluster likely to be a campaign). For example, the blog *Dancho Danchev* first report 12 clusters, each time involving 45 IOCs on average, which later also showed up on other blogs.

Table 9 shows the average percentage of IOCs first reported among all the IOCs finally discovered from a cluster (i.e., the number of first-reported IOCs and those only reported once vs. the total number of IOCs in a cluster). We found that most IOCs reported first by *Hexacorn* and *Naked Security* were also mentioned by other blogs later. Also, they provide a large amount of IOCs not documented by other blogs. We observed that even though *Webroot* only has an average of 13% of the earliest-reported IOCs for the clusters we monitored, 84% of its IOCs were not reported by

other sources.

Completeness. On the other hand, the early reports often only contain a small portion of IOCs. In our study, we measured the percentages of the IOC tokens and their iocterms for different attack clusters that were included in the first report: 6 blogs reported more than 40% of the IOC tokens and 9 blogs covered more than 50% of iocterms (related to attack behavior) per cluster. We further checked the blogs whose articles give the most complete descriptions of attack clusters. Altogether, *TaoSecurity* were found to have the largest number of such articles.

Robustness. In our research, we compared the robustness of different IOC tokens, in terms of their stability across the whole period of an attack cluster (the clusters in Table 7). From the data mentioned above, we found that the name server, C&C server, registry email are the most robust indicators, which remained unchanged in 10 to 30 percent of the clusters we analyzed (see Table 9). Using such information, we further measured the blogs likely to report these tokens: the top blogs providing most of such tokens (i.e., the number of robust IOCs vs. the number of total IOCs in a cluster) are in Table 9. Interestingly, looking into the IPs of these servers, we found that many of them actually shared the same IP prefixes, which makes us believe that they might all come from a small set of malicious Autonomous Systems.

3.5 Discussion

Our study shows that iACE makes an important step toward fully automated cyber threat intelligence gathering. This is significant not only for the convenient collection of information, but also for effective analysis of such information, as demonstrated by our measurement study. With a large amount of IOCs automatically recovered from the wild and converted into a machine-readable form, their intrinsic relations can be quickly discovered and effectively utilized to counter emerging threats. For example, knowing the sharing of C&C servers across multiple attack instances could

enable the defender to disable or block the servers to stop the attacks. On the other hand, our current design is still preliminary. Here, we discuss the limitations of our systems and potential follow-up research.

Error/missing analysis. Our study shows that iACE has a high accuracy and coverage, well beyond what standard NLP techniques can achieve. However, still our technique introduces some false discoveries and misses some IOCs. These problems mostly come from the limitations of underlying tools we use and abnormal ways of presentation. Specifically, Tesseract, the optical character recognizer, is less than perfect, and its accuracy affects the outcome of our analysis. Also, the state-of-the-art dependency parser still cannot maintain its accuracy when sentences become too long. Even though iACE only works on the shortest path between an IOC and its context token, which mitigates the problem, still there are sentences too long for the parser to understand the dependencies between words correctly. Also, adding to the complication are typos: as an example, in the case that one forgets to put a space after the period, a sentence becomes stuck with the follow-up one, which could cause an error in IOC sentence identification. Another interesting observation is that in some articles, authors deliberately misspell URLs to prevent the readers from inadvertently clicking on them: e.g., changing "`http`" to "`hxxp`" or add "[]" among dot in a URL. iACE includes a list of typical obfuscation tricks to recognize such transfers. However, there are always approaches we do not recognize, making IOC tokens fall through the cracks. Furthermore, a large amount of polluted original contents of articles might also lead to false discoveries. For example, an active attacker can compromise the blog websites and inject fake IOCs into the articles, so as to trigger iACE to report false IOCs. Further effort is needed to better address these issues.

Other intelligence sources and standards. The current design of iACE is for gathering threat intelligence from technical blogs, based on the unique ways that IOCs are described. We believe that it will also work well on other equally or more formal

sources, such as white papers and other technical articles (e.g., research papers), though further study is certainly needed here. What is less clear is the technique's effectiveness on less formal sources, like technical forums (e.g., Google groups [9], SecurityFocus [13]). The writing styles there are bit different, particularly, the use of more diverse context terms and the sentences with irregular grammatical structures. Extending iACE to this setting needs further effort. Also, the intelligence sources we use to feed iACE are all English articles. Considering the intelligence sources of other languages, iACE should import new modules for language translation of context terms and re-trained dependency parser of different languages. Further, as mentioned earlier, iACE is meant to support the OpenIOC CTI model. Although there are other models such as STIX [17] and yara [22], tools exist to convert the information across these standards.

3.6 Summary

In this chapter, we present *iACE*, a novel technique for automatic extraction of IOCs from unstructured text. iACE is designed to specialize NLP techniques to threat intelligence gathering, combining the NER and RE steps together based on the unique features of IOCs and the technical articles describing them. By anchoring a sentence with putative IOC tokens and context terms, our approach can efficiently validate the correctness of these elements using their relations, through a novel application of graph similarity comparison. This simple technique is found to be highly effective, vastly outperforming the top-of-the-line industry IOC analyzer and NER tool in terms of precision and coverage. Our evaluation of over 71,000 articles released in the past 13 years further reveals intrinsic connections across hundreds of seemingly unrelated attack instances and the impacts of open-source IOCs on the defense against emerging threats, which highlights the significance of this first step toward fully automated cyber threat intelligence gathering.

CHAPTER IV

CHARACTERIZING LONG-TAIL SEO SPAM ON CLOUD WEB HOSTING SERVICES

4.1 Introduction

Long-tail Search Engine Optimization (SEO) provides an opportunity for online advertisers to target niche markets. Instead of traditional SEO that targets a single keyword or shorter keyword phrases, long-tail SEO targets longer and more specific keyword phrases that tend to be directly related to specific products and locations. For example, a furniture marketing web page using long-tail SEO might target a more specific keyword phrase "contemporary Art Decoinfluenced semicircle lounge" rather than targeting "furniture". The advantages of long-tail SEO are that there is less competition for higher search rankings and it has been shown that specific searches are far more likely to convert to sales than generic searches [76]. As with most profitable online segments, long-tailed search results are being polluted by search engine spammers that manipulate search engine results using blackhat long-tail SEO techniques.

While long-tail SEO spamming has been an ongoing issue, the emergence of cloud web hosting services, such as Amazon S3 and Google Drive, provides a new and effective platform for dispersing long-tail SEO spam. The attractiveness of cloud hosting is that it offers fast, reliable and cheap (sometimes free) hosting. In addition, they provide a domain name that is shared by many of their users. This makes it infeasible to blacklist all content from a cloud hosting provider, which causes blacklist maintainers to expend more effort to build finer grained blacklists. Figure 9 shows an example of long-tail poisoning utilizing Google Drive, in which the second search

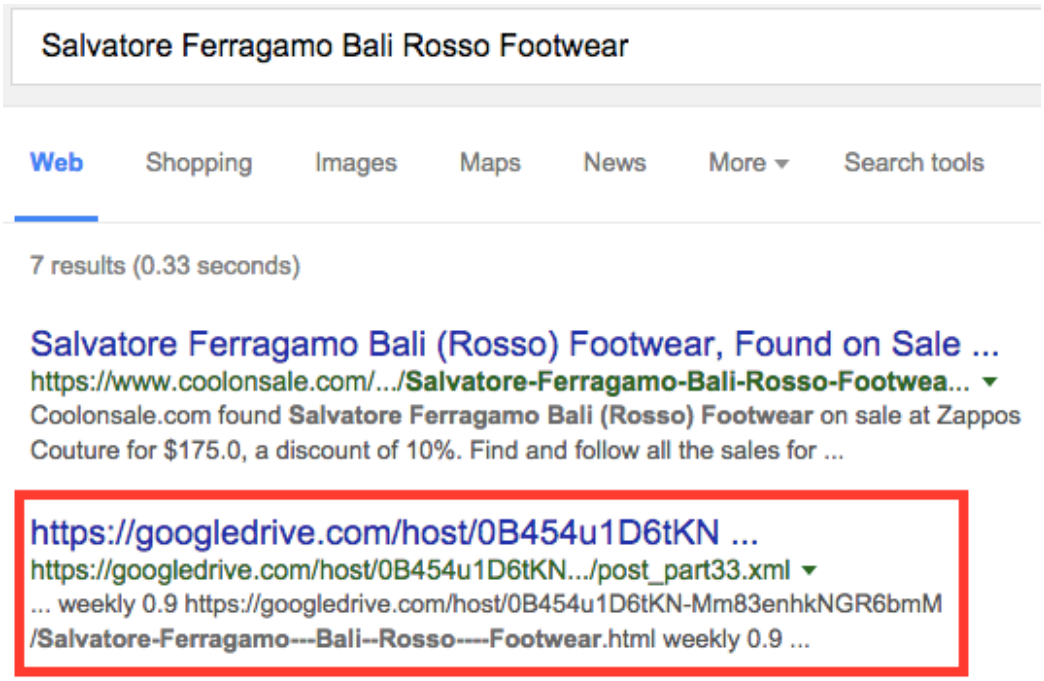


Figure 9: Example of long-tail poisoning utilizing cloud hosting platform. The second result returned here are doorway page hosting on Google's cloud hosting platform Google Drive.

result obtained from the long-tail keyword query "Salvatore Ferragamo Bali Rosso Footwear" is a doorway page with no useful content and affiliate links that are categorized as search spam by most search engines. Although there are indications of the presence of long-tail SEO spam in cloud hosting, characterizing the details of how such a spam attack is mounted, its effectiveness and spammers' ability to evade cloud platform's countermeasures have not been documented.

In this paper, we conduct the first measurement study of long-tail SEO spam hosted on cloud platforms. We bootstrapped our study by identifying spam *cloud directories* on cloud hosting platforms, in which doorway pages have largely homogeneous content in terms of their keywords and DOM structures. This enabled us to locate 930 spam cloud directories on Amazon S3 and 672 spam cloud directories on Google Drive, as well as other cloud platforms. Our analysis of the doorway pages' content revealed that they were utilizing relatively unsophisticated blackhat SEO

techniques, such as keyword stuffing (which is the repetition of keyword phrases multiple times) and keyword spam (which includes unrelated keywords). Also, we found that the SEO spammers made use of evasion techniques, such as link shorteners and obfuscated client-side JavaScript to hide affiliate links when cloud platforms do not support server-side scripting.

In order to understand the effectiveness of these long-tail SEO spam campaigns, we monitored 236,368 long-tailed keyword searches over the course of one year. Based on our analysis, we observed that 6% of the cloud-hosted doorway pages polluted the top 10 search results of long-tail keywords, and 32% of the top 100 search results. These doorway pages indicate the high-level of effectiveness of polluting long-tailed search results. We also found that almost all of the doorway pages were monetized by including links to reputable affiliate programs such as Prosperent, ClickBank and VigLink.

To understand the profitability of long-tail SEO spam on cloud hosting platforms, we analyzed the estimated revenue and click-through rate for a single campaign, which showed spammers were earning a modest sum of approximately $400 USD each per month. In addition, we noted that their click-through rates were increasing by 20% over time. Finally, we monitored ongoing interventions by the cloud service providers. We found that service providers' efforts to detect and remove doorway pages had limited effectiveness, as long-tail SEO campaigns remained active. Doorway pages on cloud hosting platforms have an average lifetime of 7 weeks, which is *much longer* than those hosted on traditional platforms (i.e., 1 week [60]).

To the best of our knowledge, our study is the first to present a comprehensive understanding of long-tail SEO spam on cloud web hosting platforms and its effects. We summarize our main contributions as follows:

1. We propose a methodology to identify cloud directories containing long-tail SEO spam, which discovered 3,186 abusive cloud directories on 10 mainstream cloud

platforms.

2. We conduct a measurement study of long-tail SEO spam on the cloud, which provides insights into its effectiveness, its use of cloud resources, network characteristics and revenue models.

3. Our empirical study shows that the cloud service provider's efforts to prevent these abusive usages are yet to be effective.

4.2 Abusive Cloud Directory Identification

In this section, we explain the methodology used in our study for abusive cloud directory identification. In the data collection stage, we first selected SEO targeted keywords to feed the search engine to identify the doorway pages on the cloud hosting service. Then, we utilized the directory structure of cloud hosting service to find other doorway pages. In the abusive cloud directory identification stage, since the long-tail SEO campaigns show high similarity in page contents in the same directories, we trained a classifier to identify the cloud directories hosting long-tail SEO spam.

4.2.1 Data Collection

In the data collection stage, we first collected the 'seed' web pages on the cloud hosting service. Specifically, we fed the SEO targeted keywords to the search engine, and used the Google Web Search API to pull the links that appeared in the search results. Second, since the web pages on the cloud platforms are organized into directories, we also crawled additional web pages in the same directories. Then, a web crawler followed the links in the page, collected their redirection chains, and stored the intermediate URL information in our local database.

Seed Data Collection. Selecting appropriate keyword phrases to feed the search engine is critical for obtaining representative results. To analyze the long-tail SEO

Table 10: List of cloud hosting platforms.

Cloud Platform	Domain
Heroku	herokuapp.com
Amazon S3	s3.amazonaws.com
Dropbox	kissr.com
Azure	azurewebsites.net
Google	googledrive.com
Openshift	rhcloud.com
Bitbucket	bitbucket.org
Sina	sinaapp.com
Baiduyun	duapp.com
Olympe	olympe.in

spam in cloud hosting services, we first choose 'hot' keyword phrases and spammy keywords phrases. These keywords reflect what people are searching for and what SEOs are targeting. Further, we use the Google Web Search API to pull the top 100 search results for each term from the Google search engine. In this paper, we analyze the long-tail SEO spam on 10 leading cloud hosting services as listed in Table 10. This set of crawled pages is defined as a seed dataset D^s, which contains 32,177 cloud URLs and 20,328 cloud directories.

For the first set of search terms, we employ popular trending keywords from Google Trend hot keywords [55]. We collect the top 20 popular search terms in 64 categories across various search interests including entertainment, education and technology. For the second set of search terms, we target some specific keywords which spammers also target. We utilized a spam trigger word list [44], which includes 200 spammy words such as "payday loan" and "casino no deposit". In addition, we gathered 20 pharmaceutical keywords, including a number of the most-prescribed and best-selling product terms from IMS Health [64]. Note that to restrict the search results to each cloud platform, we included the query "site:cloud service's domain name" (e.g., site:s3.amazonaws.com) before the aforementioned keyword phrases.

Table 11: Summary results of the datasets.

Name	# of URLs	# of cloud directo-ries	# of key-words
D^s	32,177	20,328	1,500
D^d	1,073,642	15,774	NaN

Directory Dataset Collection. On the cloud hosting service, the web pages are organized as directories. For example, a typical URL of a web page in cloud hosting service is as follows:

scheme : //dir_name.domain/file_name

where *scheme* is the protocol, e.g., HTTPS; the *dir_name* is the name of the directory shown as sub-domain; and the *file_name* is the path of the file in the cloud directory which is customized by the user. All pages from the same directory have the same *dir_name* component.

As the pages are organized as a directory in the cloud hosting service, the crawler further explores the web pages in the cloud directories which house the pages in a seed dataset D^s. Specifically, we extract the directory names from cloud URLs in the seed dataset, and then conduct another search engine query to restrict the search results to each cloud directory. Specifically, we use the keyword "site: dir_name.domain" (e.g., site:abc.s3.amazonaws.com) for the search engine query.

In this way, we generated an expanded dataset D^d, which contains 1,073,642 URLs. Ideally, the expanded dataset D^d should include all the cloud directories in the seed dataset D^s. However, as cloud platforms took action to delete the doorway pages during the course of our study, we found that 4,554 cloud directories expired. Table 11 shows the summary of the collected data.

To analyze the behavior of these cloud pages, we ran a dynamic crawler (as a Firefox add-on) to visit each cloud web page with the Referrer as google.com, and

recorded the web activities it triggered, including network request, response, and browser events. For this purpose, we deployed 20 dynamic crawlers, which were hosted on Redhat Virtual Machines with distinct IP addresses.

4.2.2 Abusive Cloud Directory Classification

Automated spam page identification on large-scale web pages is an open research question and there are no clear rules for absolute positive identification [50][86]. From the quality guidelines from Google [57], the four categories that indicate spam pages are as follows: (a) Pages generated by an automated tool or automated processes, such as Markov chains. (b) Pages optimized for a specific keyword or phrase, that then funneled users to a single destination. (c) Pages with product affiliate links on which the product descriptions and reviews are copied directly from the original merchant, without any original content or added value. (d) Pages dedicated to embedding content such as video, images, or other media from other sites without substantial added value to the visitor.

A set of heuristics were used to developed a classifier, and to detect the cloud directories used for long-tail SEO. (1) The web pages in the abusive cloud directories were optimized for a series of similar long-tail keywords. This is because to promote a targeted content, the long-tail SEO web pages utilize several long-tail keywords generated for a specific content. For example, to promote the web pages for "green coffee bean", the corresponding long-tail keywords could be "green coffee bean capsules australia", "green coffee bean capsules uk" and "green coffee bean amazon uk". (2) The web pages in the abusive cloud directories show high similarity in content and sometimes funnel visitors to the same destination websites. This is because the abusive long-tail SEO web pages are typically generated from automatic tools with a limited number of templates, and thus the web pages in the cloud directories are very similar in their DOM (i.e., document object model) structure.

Our classification began by labeling the abusive cloud directories and non-abusive directories for training. To label the cloud directories for long-tail SEO spam, we sorted the cloud directories by the number of files in the directories and manually examined the web pages. In this way, we identified 100 abusive cloud directories (10 directories on 10 cloud platforms) meeting the aforementioned definition of long-tail SEO spam. To label the non-abusive directories, we extracted the second-level domains of the URLs embedded in the cloud web pages and sorted them by their frequency of appearance. We manually examined the pages and their corresponding cloud directories with the bottom 500 second-level domains from different directories to label the non-abusive directories. Also, for those pages without an embedded URL or JavaScript, we checked if their corresponding cloud directories were non-abusive. In this way, we label 100 non-abusive cloud directories.

We extracted features from the labeled dataset in an automated fashion. Specifically, we used two sources of inputs for features: the directory features and the web pages in the directories. For the cloud directory features, we observe that the file names in the abusive cloud directories show greater similarity. This is because keywords in URLs can increase the clickthrough rate in the search engine result pages [70], and the abusive user tends to make the long-tail keywords visible in the URLs. Hence, highly similar long-tail keywords in URLs show as similar file names in the abusive cloud directories. To calculate the file names' cosine-similarity, we extract the file names from the path component of the cloud URL, and then tokenize them into words using separators such as '-' and '_'. Then, the words in each file name is converted into a sparse vector, and we calculate cosine-similarity for the vectors in the same cloud directories.

For the web page in the directories, the main reason we extract features from the raw HTML is that long-tail doorway pages in the abusive cloud directories shows

high similarity in page content, such as meta keywords, page title and page template because of automatic page generation. To extract HTML source features, we follow a conventional n-gram approach. Particularly, we choose to build 3-gram features. The rationale is that a 3-gram can capture the structure for a sequence (e.g., `affid=12345`) very well. Each Meta keyword, URL and script in the web page is segmented into words so that each word is either one of the reserved characters in `‘! * ’ ( ) ; : @ & = + \$ , / ? \% # [ ] ’` , or contains no reserved characters. We convert each word into a sparse vector with the dimensions of the same number of 3-grams. On each dimension, the value is proportional to the frequency of the corresponding n-gram. Each vector is normalized to have the L_1 norm [93].

Subsequently, we trained a SVM (i.e., support vector machine) [93] classifier over the training set. We evaluated the predictive accuracy of the classifier by performing 10-fold cross-validation on the labeled dataset, yielding a 92% rate of successful classification. In the end, the algorithm classified 3,186 abusive cloud directories. To validate these predictions, we manually inspected additional subsets of unlabeled examples. Without loss of generality, we utilize Chernoff Bounds [84] to estimate the number of pages to be sampled. We set the trust interval $\delta = 0.01$ and the error probability $\lambda = 0.01$ to obtain the number of sampled cloud directories $n = 500$. After manually inspecting the sampled cloud directories, we find that around 12 of the cloud directories are false positives which is consistent with the predicted 92% rate.

4.2.3 Ethical concerns

In order to avoid unintentionally advertising for abusive actors, we do not include the actual names of abusive cloud directories and vendors. Instead of including the raw URL of spam directories and doorway pages, we adopt the naming convention of

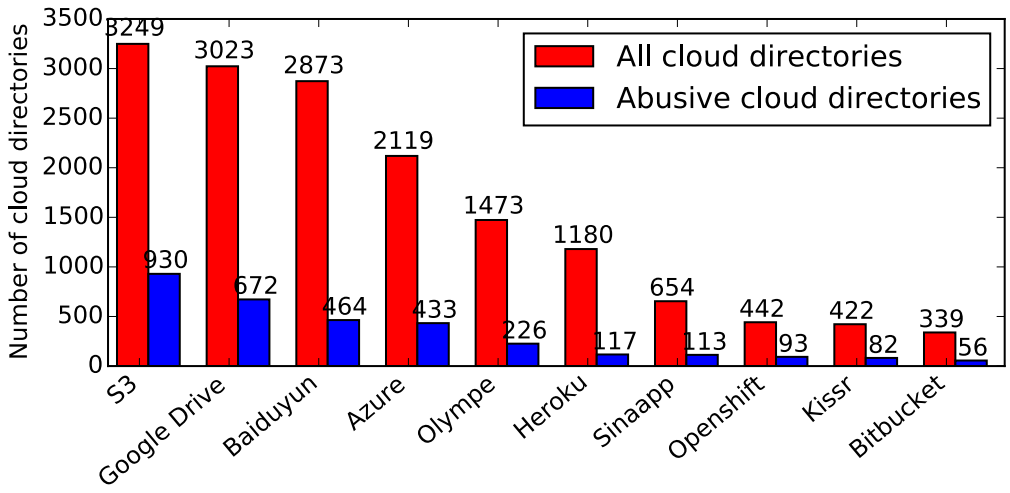


Figure 10: Number of abusive cloud directories on each cloud platform.

`<cloud provider>_<affiliate program+number>` to minimize the impact on privacy. When including content from these doorway pages we redact all raw identifiers, such as URLs, identifying comments and other potentially identifying information. Also, we limit our analysis to public URLs that are indexed by a search engine for identification and measurement. We did not try to access the base directory listings in order to minimize the impact on privacy.

4.3 Long-tail SEO on the cloud

In this section, we study the effectiveness of long-tail SEO spam on cloud web hosting services, i.e., the prevalence of long-tail SEO spam on cloud web hosting as well as their impact on organic long-tail keywords search results. We found that 6% of the long-tail SEO doorway pages we observed successfully poisoned the top-10 search results for long-tail keywords included in our study. Then, we provide a perspective of the blackhat SEO techniques and the evasion techniques the abusive user adapted for the cloud web hosting platforms.

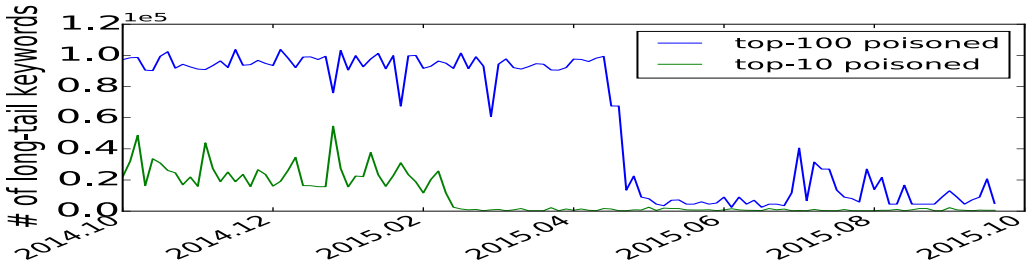
Overview. We start by discussing the prevalence of abusive cloud directories for long-tail SEO spam on cloud web hosting platforms. Of the 15,774 cloud directories

we collected, we found that 3,186 directories (318,470 doorway pages) were long-tail SEO spam.

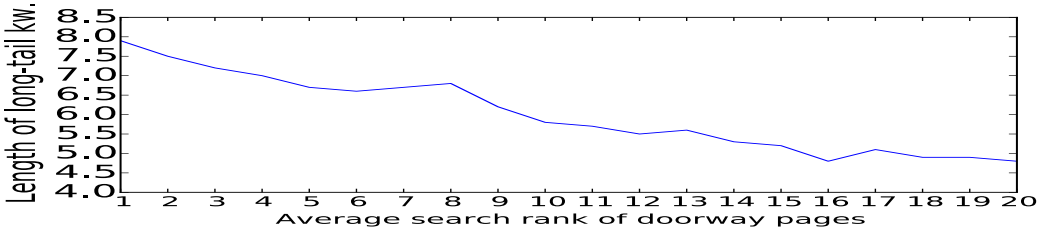
Figure 10 illustrates the number of abusive cloud directories on each cloud platforms. Among them, Amazon S3 is the most popular (28%) in our dataset, followed by Google Drive (22%). The result shows that the abusive cloud directories for long-tail SEO is being hosted on cloud platforms. Note that of these 10 cloud platforms, eight of them provide free hosting services (e.g., 5GB for Amazon S3, 15GB for Google Drive), and therefore are ideal platforms for low-budget abusive users. These users also take advantage of the pay-as-you-go feature of cloud hosting to conduct low cost long-tail SEO, which does not require traditional SEO back linking techniques [72][73]. Lastly, long-tail SEO pages hosted on the cloud are more difficult to blacklist since cloud hosting domains also host a large amount of benign content.

Effectiveness of Long-tail SEO. To analyze the search engine poisoning impact of long-tail SEO spam on the cloud, we extracted 236,368 distinct long-tail keywords from doorway pages in the abusive cloud directories we identify, and then crawled the top 100 organic Google search results of the long-tail keywords from 10/2014 to 10/2015.

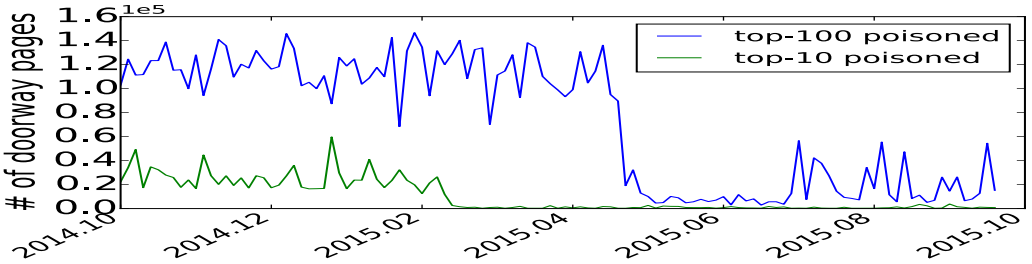
To extract the keywords, we implemented a stuffed keyword extraction tool based on n-grams. We define an N-gram as a contiguous sequence of n words in the HTML files. First, we extract the text from the DOM tree using an open-source tool `BeautifulSoup` and use white space as the token separator. Then, we calculate the frequency of each n-gram. In our implementation, we set the range of n from 3 to the length of page title l. After that, we compared the n-gram tokens' frequencies f where $n \in [3, l]$ and used the n-gram token with the largest keyword density $d = \frac{n \times f}{T}$ as the stuffed keywords, where n is the length of the keyword token, f is its frequency and T is the number of words in a page.



(a) **Evolution of number of poisoned long-tail keyword.**



(b) **Long-tail poisoned keyword length distribution.**



(c) **Evolution of number of doorway pages.**

Figure 11: Effectiveness of Long-tail SEO spam.

For example, the owner of the abusive cloud directories on Google Drive uploaded a keyword stuffing doorway page '1403682103503-aclarar-la-piel-para-siempre—oficial.html' with 775 word phrases. The page has the largest 3-gram token 'la piel para' with frequency 47, largest 4-gram token 'la piel para siempre' with frequency 42, largest 5-gram token 'aclarar la piel para siempre' with frequency 35 and largest 6-gram token 'aclarar la piel para siempre oficial' with frequency 12. The stuffed keyword extraction tool will extract the long-tail keyword 'aclarar la piel para siempre' with the largest percentage 22.5%.

Surprisingly, we found that the doorway pages in the abusive cloud directories successfully poisoned the highly specific long-tail keyword phrases. Figure 11(a) illustrates the evolution of the number of poisoned long-tail keywords over time. We

define the long-tail keywords as poisoned if the abusive cloud directories appeared in the top 10 (i.e., indicated as top-10 poisoned in figures) or top 100 (i.e., indicated as top-100 poisoned in figures) organic search results. During the period from 10/2014.10 to 2/2015, 9% of the long-tail keywords were poisoned in the top 10 organic search results. This number jumped to 42% for top 100. In general, the trend exhibits a substantial decrease in the number of poisoned keywords, because cloud providers will remove the doorway pages. We also observe that non-English keywords were easier to be poisoned, such as 'como pintar con oleo', which has the relevant doorway page ranked as the first search result.

Figure 11(b) illustrates the average length of the poisoned long-tail keywords in search rank from 1 to 20. Overall, the average length of poisoned keywords increases while the search ranks of doorway pages become higher. This is because the shorter keywords have higher competition and is therefore difficult to be polluted. The average length of the keywords, whose corresponding doorway page's poisoned search rank is 1, is around eight. However, when the keyword length is 6, the average search rank of doorway pages decreases to 10.

Figure 11(c) shows the evolution of the number of doorway pages we found in the top 10 and top 100 organic search results for the poisoned long-tail keywords. On average, 6% of the doorway pages are ranked in the top 10, which is 32% in top 100. From Figure 11(c), we can see that the prevalence of the doorway pages in the organic search results. For an example of SEO effectiveness, 100 doorway pages in the abusive cloud directories `googledrive_markethealth` successfully poisoned 61 long-tail keywords' top 100 search results, which will redirect the visitors to the same online pharmacy vendor, a site that was reported as a scam website by reviewopedia [98]. Among the 61 poisoned keywords, the doorway pages appeared in 5 long-tail keywords' top 5 search results. Examples of the poisoned long-tail keywords include 'green coffee bean diet does it work' and 'green coffee bean cleanse australia'.

Blackhat SEO technique. We examined the blackhat SEO technique that the spam campaigns utilized to poison search results. Our research surprisingly revealed that using simple blackhat SEO technique (e.g., keyword stuffing), doorway pages were able to successfully poison the search results. In addition to blackhat SEO techniques, such as keyword stuffing and social fraud, targeted blackhat SEO techniques were also used, incorporating multiple cloud provider related elements such as adding products as unrelated keywords or misleading visitors by adding the cloud provider's logo.

Keyword poisoning is the deliberate manipulation of the search engine's index for specific keyword terms. It involves a number of methods such as keyword stuffing (i.e., the repetition of keywords in the meta tag and page contents), and traffic spam (i.e., adding unrelated keywords to manipulate the relevance).

Regarding the doorway pages' keyword densities that we obtain from Section 4.3, 84% of doorway pages have a keyword density larger than 15%, which is less than 3% for web pages in non-abusive cloud directories that we mention in Section 4.2. As an example of keyword stuffing, in the doorway pages uploaded in the abusive cloud directory `googledrive_clickbank`, keywords were repeated multiple times in the content of the pages. To hide the stuffed keywords from human readers, abusive users set white text on a white background or located the stuffed keywords behind figures in the doorway pages.

To measure the keywords relevance to identify traffic spam, we studied the doorway pages with more than one META keywords. We extract the keywords from the META tag of the doorway pages and query their semantic similarity using DISCO API. If the keywords have a large semantic gap (semantic similarity¡0.05), we determine that the doorway page utilizes traffic spam techniques. Using this method we find that 48,922 doorway pages in 526 abusive cloud directories utilize traffic spam techniques to manipulate the page relevance. Interestingly, the abusive users include

cloud platform-related information as the stuffed keywords or unrelated keywords, such as `Google Plus` and `Youtube`. For example, the doorway pages that masquerade as an online flower shop utilize "Proflowers Google Plus" or "lotus flower youtube" as the keyword phrases. We observed that 16% of the doorway pages on Google Drive use Google product terms as unrelated keywords, which is 5% on Amazon S3.

Evasion technique. Given the prevalence of the doorway pages on cloud hosting platforms, we examine the evasion techniques used to avoid detection. We found that the spam campaigns adapted evasion techniques for cloud web hosting platform, such as link shorteners and obfuscated client-side JavaScript when cloud platforms do not support server-side scripting.

As the illicit practices of doorway pages and manipulating search rankings can lead to the pages being removed from the Google index [57], the attackers utilize evasion mechanisms to avoid detection. However, as most of the cloud hosting platforms (e.g., Google Drive, Amazon S3) do not support server-side scripting and a simple client-side script for evasion is easily detected, abusive users make several changes to adapt to the cloud web hosting platform. Many evasion mechanisms were used by the abusive users, such as obfuscation, link shortening and redirection cloaking.

1) Mixed redirect cloaking. Cloaking refers to deceiving search engines by providing different content to the search engine crawlers compared to users clicking on search results. Cloaking on the traditional platform includes client-side cloaking (e.g., use client-side scripting to store cookies) and server-side cloaking (e.g., using server-side scripting to track IP). Compared to client-side cloaking, server-side cloaking is more concealed and much more likely to circumvent detection [118]. As most of the cloud hosting platforms do not support server-side scripting, we observed mixed redirection cloaking, which combined the client-side cloaking on doorway pages and server-side cloaking on the external server. The abusive user utilized mixed redirection cloaking

```
<script type="text/javascript">
       if (document.referrer != ""){
              var refer_url = document.referrer;
              var post_url = "http://www.gatherguideshare.info/product/" + data_loc;
              document.write('<form name="form1" method="post" action="' + post_url + '"><input name="info" type="hidden" value="' + data_info + '" /><input na
              document.form1.submit();
       }
</script>}
```

Figure 12: Redirection cloaking used by amazon_gatguisha-20

as shown in Figure 12. When the user visits the webpage, a POST request is dynamically generated by the Javascript implementation to report the `document.referrer` to the external server `gatherguideshare.info`. The external sever then operates the server-side redirection cloaking based on the `document.referrer`, i.e., the external sever will respond with status code 302 for the POST request to redirect normal visitors (e.g., those who visit doorway pages by clicking through search engine results) to `gatherguideshare.info`, while search engine crawlers receive content crafted to rank well for targeted query terms (e.g., "compaq armada dock station").

2) Obfuscation. Obfuscation is the deliberate act of creating code that is difficult for humans to understand. Obfuscation, as another way to circumvent static analysis of the client-side illicit script, is also widely used in the doorway pages on cloud platforms. For example, the redirection cloaking code we mentioned in Figure 12 was obfuscated by the character code. Note that by combining the cloaking and obfuscation techniques, the doorway pages from `s3.amazonaws.com_gatguisha-20` have a longer lifetime (more than 15 weeks we observed) than other doorway pages on the same cloud platform (average 7 weeks, detailed in Section 4.5). Other forms of obfuscation were also found, such as word substitution, which separates key phrases (e.g., campaign ID) into fragments with random order.

URL shortening is another evasion technique used by the abusive users to circumvent static analysis from the cloud service provider. For the doorway pages in cloud directory `googledrive_filepost`, a shortened URL was generated dynamically by the Javascript code, which requests bit.ly URL shorten API `https://api-`

Table 12: Top 5 affiliate networks where most abusive cloud directories belong to.

Affiliate network	# of doorway pages	# of cloud directories	Volume
Amazon	7,663	72	2.4%
viglink	6,272	64	2.0%
prosperent	5,077	52	1.6%
Clickbank	4,689	51	1.4%
MarketHealth	4,177	43	1.3%

`ssl.bitly.com/v3/shorten` for each doorway pages. In the Javascript implementation, a long URL was first generated with the parameter in "asin" tag of each page, and the fixed domain and path. Then, a bitly URL shortener API was called to return the shortened URL for the original one. Note that the fixed domain and path were also obfuscated by the BASE64 code and the shortened URL was generated at run time.

In addition to `bit.ly`, multiple URL shorteners are utilized by the abusive users such as `t.co` (0.6% of doorway pages), `goo.gl` (1.5% of doorway pages) and `tinyurl.com` (5% of doorway pages).

4.4 Traffic Monetization

In this section, we study how the long-tail SEO campaigns monetize traffic. We find that almost all of the long-tail spam campaigns are monetized by sending visitors to affiliate programs. Further, we identify five large long-tail spam campaigns and surprisingly find that they are mainly working for reputable affiliate networks (e.g., prosperent.com) or for reputable online vendors' promotion (e.g., Amazon). Traffic monetization techniques used by the long-tail spam campaigns were analyzed, followed by a revenue analysis.

Table 13: Example of regular expressions for campaign ID identification.

Campaign ID Regex
prosperent.com/store/product/[0-9]{6}-[0-9]{4}-[0-9]
redirectingat.com?id=[0-9]{5}X[0-9]{7}
\w.\w.hop.clickbank.net
247rxshop.com/?affid=[0-9]{8}
paydaylendersearch.com/[a-z0-9]{8}-[a-z0-9]{4}-[a-z0-9]{4}-[a-z0-9]{4}-[a-z0-9]{12}

Affiliate structure of long-tail spam campaigns. Different from traditional platforms, doorway pages on cloud platforms share a common second-level domain and trusted name servers belonging to cloud platforms. Thus, to look at the network topology, we built network topology graphs G_{ta} for the abusive cloud directories. In the graphs, each IP of the redirectors and landing servers is regarded as a node, and the abusive cloud directories are set as the starting nodes. Each edge corresponds to a redirection between two nodes.

We manually review the network topology G_{ta} of abusive cloud directories and found that many of them were organized as affiliate programs. For the graph G_{ta} with 6,012 nodes and 47,398 edges, we surprisingly only found that *the long-tail SEO campaigns on the cloud web hosting platform show great connectivity in network topology.* As hubs in the graph G_{ta}, the top 3 nodes with the largest in-degree in G_{ta} are `amazon.com`, `prosperent.com` and `viglink.com`. The top three nodes with the largest out-degree in G_{ta} are `clickbank.net`, `redirectingat.com` and `dotomi.com`. By reverse DNS lookup, we found that *each of them belongs to reputable affiliate networks.*

Next, we utilize the hubs in the graph to give an overview of the affiliate structure of the abusive cloud directories. Table 12 provides an overview of the top 5 affiliate networks that host the most abusive cloud directories. Overall, the majority of the abusive cloud directories come from abusive users that work for reputable affiliate networks. These affiliate networks mostly collaborate with well-known online vendors that have a policy, based on abuse reports from individuals, to prohibit their

affiliates from using the service in conjunction with network abuse or spam. We observe that the largest amount (2.4%) of doorway pages belong to 72 abusive cloud directories working for the reputable affiliate network `Amazon`. Thus, it appears that even though most of the reputable affiliate networks have policies that govern the affiliates to prevent abuse and search engine attacks, illicit practices are still found in these reputable affiliate networks.

Identifying the affiliate ID of the abusive users would help to identify spam campaigns, prevent their spread and efficiently remove the doorway pages. Our idea is to extract the affiliate IDs from the redirection chains of the doorway pages. To do so, we designed a semi-automatic common substring-based algorithm to generate regular expressions to extract affiliate IDs, as follows: (1) We extract a common string from each directory by the cross-comparison between the redirection chain inner pages and the redirection chains among the pages in the directory using a generalized suffix tree [61]. Note that we consider each order of the parameters in the URL query string, i.e., for both cases `example.com/?a=1&b=2` and `example.com/?b=2&a=1`, we calculate their common strings. (2) We generate the regular expressions by mapping the digits and English alphabets in the URL parameter into formal language. (3) We manually check the correctness of these regular expressions such as accessing the affiliate network for marketing URL information and manually inspecting the sampled pages. In this way, we labeled 2,360 abusive cloud directories with 342 affiliate IDs (a.k.a., abusive entities), which were associated with 225,008 long-tail SEO doorway pages. We present the cumulative distribution of the number of cloud directories per abusive entities in Figure 13. Figure 13 shows that 80% of the abusive entities are associated with more than one cloud directory. Moreover, 14% of abusive entities distribute doorway pages on different cloud platforms. This might be because the abusive entities are concerned about being detected by cloud platforms, and so they distribute doorway pages into different directories, and cloud platforms.

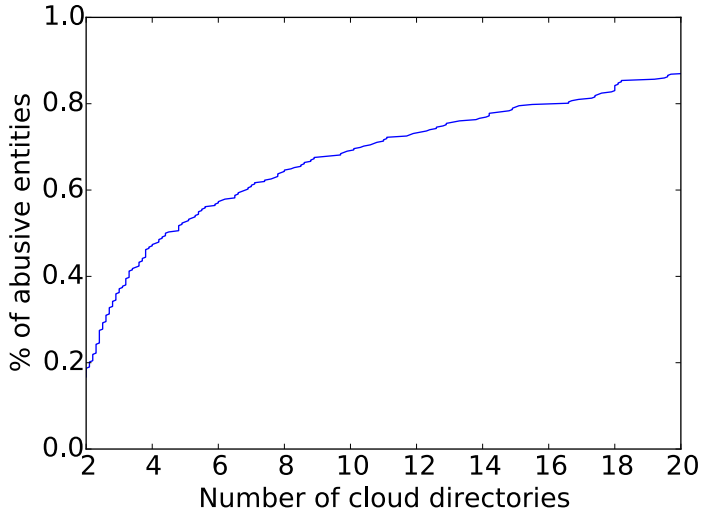


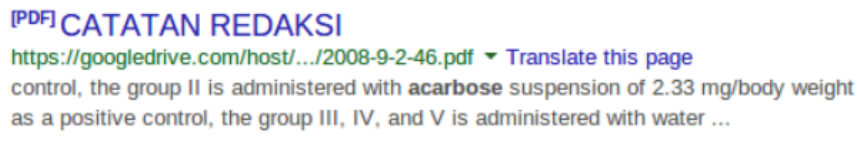
Figure 13: Cumulative distribution of number of cloud directories per abusive entities.

Dissecting traffic monetization techniques. Long-tail spam campaigns monetize traffic through multiple vectors, such as search redirection and social fraud.

Search redirection. The search redirection technique has been regarded as the blackhat technique to lure traffic. When visitors click the link in the search engine result pages, they will be redirected to a different site rather than the one pointed to by the link. Traditional search redirection attackers prompt a site server to conduct request redirection via code injection. On the cloud platform, the abusive users utilize client-side script to redirect visitors, such as iframe, JavaScript (e.g., windows.location) and POST request.

We determine the search redirection with the dynamic crawler (see Section 4.2), and further analyze the semantics consistency of the source page and the landing page. Specifically, we utilized Yahoo content analysis API [124] to extract a series of keywords from the source page and the landing page. If the keyword sets did not intersect, the search redirection shows semantic inconsistency.

In this way, we find that 63,900 doorway pages in 769 abusive cloud directories utilized search redirection to monetize traffic. Among them, 23% of the doorway pages




```
1  <div xmlns:v="http://rdf.data-vocabulary.org/#" typeof="v:Review-aggregate">
2    <p align="center">
3      <span rel="v:rating"><span typeof="v:Rating"><span property="v:average"
           >8</span>of<span property="v:best">10</span> </span> </span> based
           on<span property="v:votes">11</span> Ratings. <span property="v:
           count">27</span>reviews
4    </p>
5  </div>
```

Figure 14: Fake product score shown in the search result page.

redirected visitors to a semantic inconsistency landing page. For example, 280 doorway pages were uploaded in the abusive cloud directory `googledrive.com_mediaupdate41` for malware distribution. The doorway pages masquerade as web pages that sell flowers to funnel visitors to a malware distribution website.

Social fraud. Social fraud techniques mislead Google users by manipulating the search snippets. Google's Rich Snippets technique [56] allows users to summarize the content of a page such as a product's review. Rich Snippets help visitors recognize the relevancy of their search and trigger potential clicking. However, Rich Snippets can be directly inserted in the page without validation. Abusive users can thus leverage this technique to provide fake review scores or irrelevant reviews. For example, the doorway page in the abusive cloud directories `s3.amazonaws.com_markethealth` makes up irrelevant reviews using rich snippets which shows in the search result to attract clicks. Figure 14 shows Rich Snippets that have been abused.

Revenue Analysis. To understand the economic motives behind the abusive activities, we analyzed the revenue received by these users. We utilize the following

revenue model which was proposed in prior research[31][86]: $R(t) = N_v(t) \cdot P_a \cdot R_a$, where the total revenue $R(t)$ during the time period t is calculated from the total number of actions taken (i.e., click-through number [86], $N_v(t) \cdot P_a$) and the average revenue per action R_a.

To investigate the increase rate of the click-through number (i.e., $N_v(t) \cdot P_a$), we track the number of URL clicks for the doorway pages uploaded by the abusive user who works for a shady affiliate network called *filepost.ml*. The affiliate network *filepost.ml* hires affiliates to promote its fake free e-book download website which lures visitors to finish many cost-per-action affiliate programs. The abusive user hosts 384 doorway pages in one cloud directory and hides the marketing URL by using the URL shortener `bitly.com`. As `Bitly` provides an API to count the number of clicks for its shortened URL, we obtain the click number of the marketing URL in the abusive doorway pages.

Figure 15 shows the cumulative click-through number of the 384 doorway pages from Sep. 5, 2014 to Feb. 18, 2015 . Hosting the 384 doorway pages on Google Drive, the abusive entity will see around a 1,800 click increase every month. The click increase rate is around 20% per month. Utilizing the same revenue model and parameter setting $R_a = \$0.265$ as prior works [31][86], we can estimate the revenue for `googledrive.com_filepost` in October 2014 of $R(1\ month) = (5249 - 3754) \times 0.265 =$ \$396, which increased to \$665 in January 2015. Note that with the evasion technique we mentioned in Section 4.3, abusive cloud directories have extremely long lifetimes (i.e., more than 40 weeks), which helps the abusive users gain more profit.

4.5 Intervention

In this section, we monitored ongoing interventions by the cloud service providers. Since the abusive users violate the usage policies of cloud platforms [57] and poison search engine results to degrade users' experience, cloud providers tend to remove

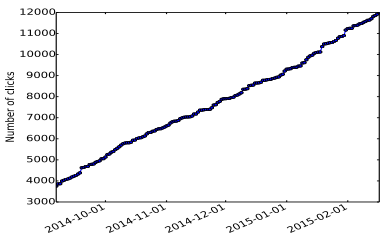


Figure 15: The cumulative number of clicks of the doorway pages on the abusive cloud directory `googledrive.com_filepost`.

doorway pages entirely from the cloud platforms. However, as our empirical analysis shows, these cloud providers' efforts are far from effective.

To measure the average lifetime of the doorway pages and the abusive cloud directories, we re-crawled the active doorway pages every three days and used the lifetime as the time between the first and last time the crawler observed a page. Figure 16 illustrates the percentage of doorway pages and abusive cloud directories in different lifetime ranges. We found that the average lifetime of the doorway pages is around seven weeks, which is much longer than those hosted on the compromised sites (i.e., around one week [60]). Moreover, the average lifetime of the abusive cloud directories was extremely long (around 20 weeks). In the empirical study, we observed that though the cloud providers found and removed the doorway pages, they did not aggressively remove the corresponding abusive directories, or the doorway pages from the same abusive entities in different cloud directories.

Then, we analyzed the abuse situation of doorway pages in the cloud web hosting platform, i.e., the evolution of the newly-appeared doorway pages and abusive cloud directories. Hence, we resubmitted the hot and shady keywords to the search engine every three days from 2014.10 to 2015.10. At each measurement point, we crawled the data in the same way as mentioned in Section 4.2. In this way, we built the time-period dataset D^t. At each measurement point, the average number of URLs we crawled was around 500K associated with 3K cloud directories.

Figure 17(a) shows the evolution of the number of newly-appearing abusive doorway pages, compared with the number of deleted doorway pages we found in Section

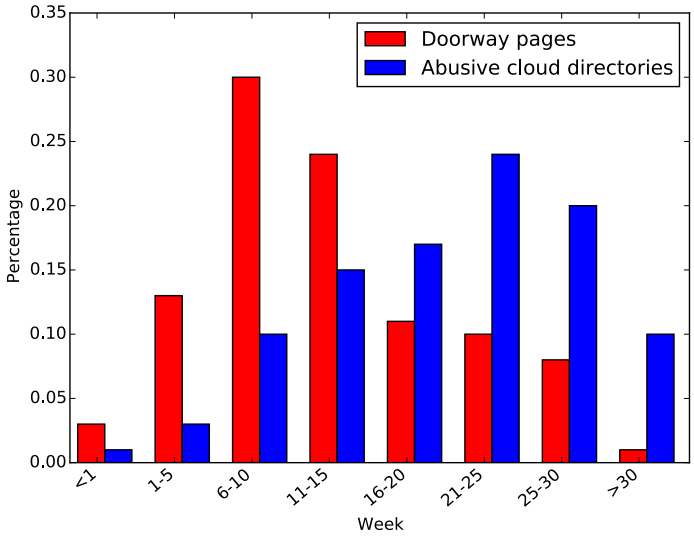


Figure 16: Lifetime of doorway pages and abusive cloud directories.

4.2. The evolution of the abusive cloud directories is shown in Figure 17(b). From 17(a), we can observe that large amounts of doorway pages newly appear, which has a higher rate of increase than deletion rate by the cloud provider. Also, 23% of the newly-appeared doorway pages were associated with the known abusive directories, and the rest of them belonged to the newly-appeared abusive directories. Also, from Figure 17(b), we observed that the deletion rate of the abusive cloud directories is much smaller than that of doorway pages. This shows that the detection method used by cloud platform did not identify a large enough amount of doorway pages for each abusive cloud directory or remove the abusive cloud directories. Moreover, we observe an increased deletion rate from 2014.12 to 2015.02, because the cloud provider *Google Drive* took more efficient action to remove doorway pages. As the doorway pages can be easily spread on the cloud web hosting platform, the abuse situation will become worse if the detection method is not effective enough.

Figure 17(c) shows the prevalence of doorway pages for three abusive cloud campaigns. The trend line shows the number of doorway pages in the 1,520 'hot' keywords top 10 search results restricted to their cloud platform. For the three abusive campaigns, the number of doorway pages appeared in the top 10 search results did not

change much in October and November. For the abusive user `amazon_beslca0e-20`, more doorway pages poisoned the top 10 search results in November. Figure 17(d) shows the number of doorway pages that appeared in the deleted page set over time. Even though the deleted doorway pages will be removed from the top 10 search results, the active doorway pages from the same abusive cloud directories stay at the same measurement point, which means that cloud provider did not detect and remove all the doorway pages from the same campaign.

4.6 Discussion

In this section, we discuss the limitations of our study and potential mitigation strategies.

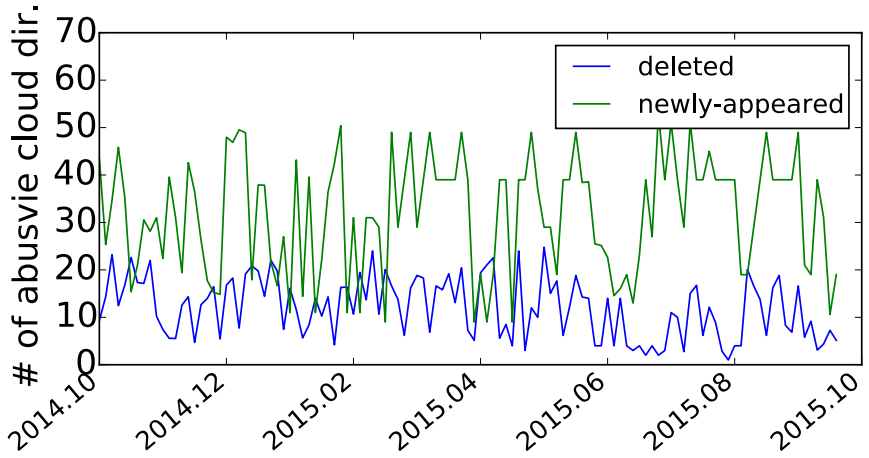
Limitation As mentioned earlier, long-tail SEO spam identification on large-scale cloud data is difficult, especially for a third party. Our design has a number of limitations imposed by our vantage point, over-restricted features and manual validation. First, our methodologies' vantage points are limited to Google's search results. While Google is the mainstream search engine targeted for search engine poisoning, the results we crawled were limited to the cloud web pages that are indexed by Google. Second, the insight for feature extraction is that the abusive user tends to put several doorway pages in cloud directories for long-tail SEO spam, and the doorway pages are auto-generated and hence show similarity. While this insight was validated by our pre-measurement study on training data, there may be small numbers of abusive users intentionally increasing the keyword and DOM source diversity to evade detection. Hence, the over-restricted feature design may bias our technique to low false positives but relatively smaller coverage. Third, we use manual inspection to validate abusive cloud directories, which is laborious and may include false positive.

Mitigation strategies Based on the results of our measurement study, we have identified several potentially effective mitigation strategies to reduce the impact of long-tail SEO spam on the cloud hosting platforms. First, search engines, could detect the highly similar low-quality content of these doorway pages and penalize them in the search rankings. This would cause these spammers to expend more resources creating less duplicated, higher quality content. Second, the cloud providers could increase the cost of establishing accounts on their services and more aggressively detect and remove spammy cloud hosting accounts. While this can result in an escalating detection and evasion arms-race, our analysis of these doorway pages found that identifying affiliate IDs can be done automatically and these can be used to detect and remove large numbers of accounts hosting doorway pages. Finally, the affiliate networks could monitor HTTP refers and identify other indications that their affiliates are engaging in SEO spam. We found that most of the affiliate networks currently have reactive policies, such as abuse reporting to restrict illicit practices of affiliates. A more proactive policy might help to mitigate the surge of long-tail SEO spam on cloud hosting platforms.

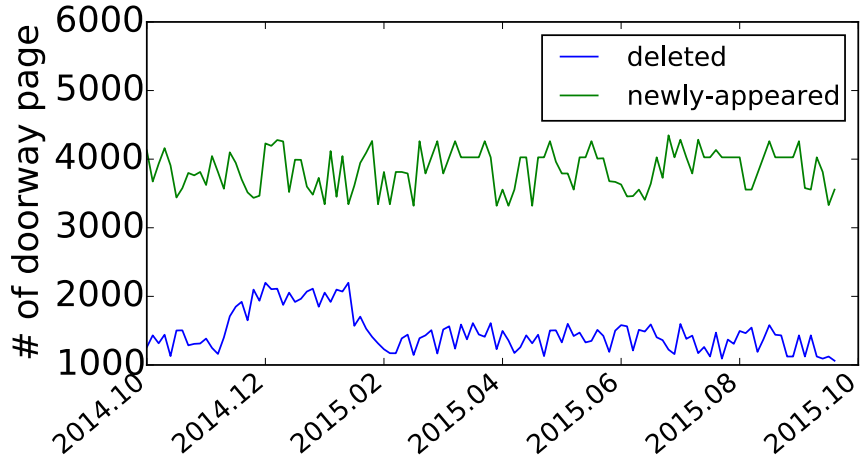
4.7 Summary

To the best of our knowledge, a comprehensive overview of the long-tail SEO spam on the cloud web hosting platform, and measurement study of the abusive activities are still open research challenges. In this chapter, we conduct the first study to measure, and analyze the long-tail SEO spam on cloud web hosting platforms. Specifically, we identified 3,186 abusive cloud directories for long-tail SEO spam from analyzing approximately 15,774 cloud directories over 10 cloud platforms. Then, we conducted an in-depth measurement study of the abusive cloud directories for long-tail SEO spam. As a result of our measurement, we uncover that the abusive users take advantage of the pay-as-you-go feature of cloud hosting to conduct low cost long-tail

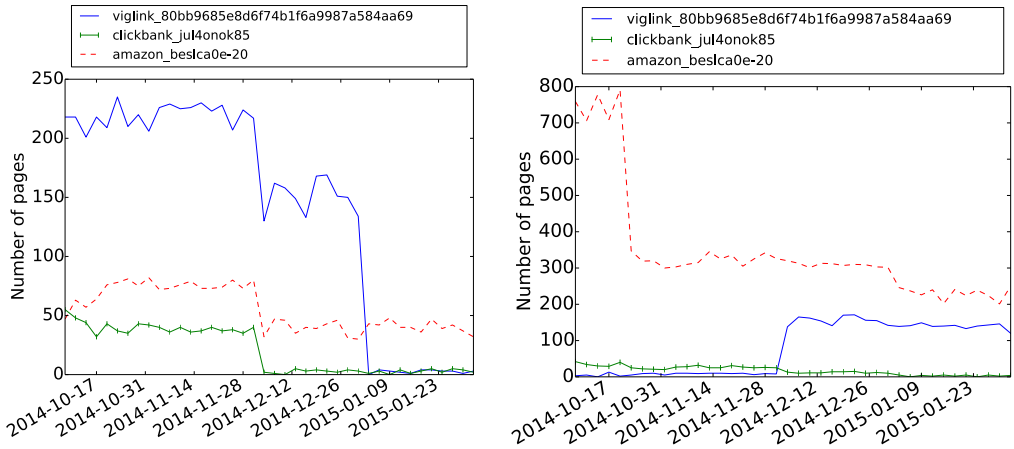
SEO. Our measurement study provides insights into long-tail SEO spam effectiveness, blackhat SEO techniques they used, and network characteristics of the long-tail SEO campaigns. Moreover, the intervention of the cloud provider is analyzed, which is shown to be far from effective. Our findings for the long-tail SEO spam on cloud hosting platforms enable us to deeply understand the abusive long-tail SEO spam, which enable an important step toward effective mitigating of this new type of security threat.



(a) Number of abusive cloud directories over time.



(b) Number of doorway pages over time.



(c) Number of active doorway pages in Top 10/100 search ranking per measurement point.

(d) Number of deleted doorway pages in Top 10/100 search ranking per measurement point.

Figure 17: The increasing trends of active pages, deleted pages and doorway pages over time are shown in Figure 17(a). The corresponding accounts are shown in Figure 17(b). Number of doorway pages for the three campaigns over time are shown in Figure 17(c) and 17(d).

CHAPTER V

UNDERSTANDING CLOUD REPOSITORY AS A MALICIOUS SERVICE

5.1 Introduction

Cloud hosting service today is serving over a billion users world-wide, providing them stable, low-cost, reliable, high-speed and globally available resource access. For example, Amazon Simple Storage Service (S3) is reported to store over 2 trillion objects for web and image hosting, system backup, etc. In addition to storing data, these services are moving toward a more active role in supporting their customers' computing missions, through sharing the repositories (a.k.a. *bucket* for Google Cloud [4]) hosting various dynamic content and programming tools. A prominent example is Google's Hosted Libraries [58], a content distribution network (CDN) for disseminating the most popular, open-source JavaScript resources, which web developers can easily incorporate into their websites through a simple code snippet. In addition to benign users, the popularity of these services has also attracted cybercriminals. Compared with dedicated underground hosting services, repositories on legitimate commercial clouds are more reliable and harder to blacklist. They are also much cheaper: for example, it is reported that 15 GB on the dark net is sold at $15 per month [14], which is actually offered for free by Google to every Google Driver user. Indeed, it has been reported [108] that malware distributors are increasingly using the commercial clouds to process and deploy malicious content.

Understanding bad cloud repositories: challenges. Although there have been indications of cloud hosting misuse, understanding how such services are abused is challenging. For the service providers, who are bound by their privacy commitments

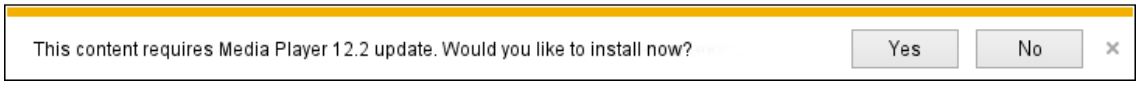


Figure 18: Example of deceptive images in Amazon S3 bucket *cicloudfront* used for malvertising. The image was shown at the bottom of a webpage as an update notification to lure visitors to download malware.

and ethical concerns, they tend to avoid inspecting the content of their customers' repositories in the absence of proper consent. Even when the providers are willing to do so, determining whether a repository involves malicious content is by no means trivial: nuts and bolts for malicious activities could appear perfectly innocent before they are assembled into an attack machine; examples include image files for Spam and Phishing as shown in Figure 18. Actually, even for the repository confirmed to serve malicious content like malware, today's cloud providers tend to only remove that specific content, instead of terminating the whole account, to avoid collateral damage (e.g., compromised legitimate repositories). Exploring the issue becomes even more difficult for the third party, who does not have the ability to directly observe the repositories and can only access them through the websites or sources that utilize the storage services. Further adding to the complexity of finding such a repository is the diverse roles it may play in attack infrastructures (e.g., serving malware for one attack and serving Phishing content for another), due to the mixed content a single repository may host: e.g., malware together with Phishing images. As a result, existing techniques (e.g., those for detecting dedicated malicious services [78][89]) cannot be directly applied to capture the repository, simply because their original targets often contain more homogeneous content (e.g., just malware) and contribute to different campaigns in the same way. So far, little has been done to understand the scope and magnitude of malicious or compromised repositories on legitimate clouds (called *Bad Repository* or simply *Bar* in our research) and the technical details about their services to the adversary, not to mention any effort to mitigate the threat they pose.

Finding "Bars" online. In this paper, we present the first systematic study on the abuses of cloud repositories on the legitimate cloud platforms as a malicious service, which was found to be highly pervasive, acting as a backbone for large-scale malicious web campaigns (Section 5.3). Our study was bootstrapped by a set of "seeds": 100 confirmed malicious or compromised buckets [4], each of which is a cloud resource repository with stored objects (often of different types) organized under a unique identification key. These buckets were collected from Spam messages or the malicious URLs cached by a popular malware scanner. Comparing them with those known to be legitimate, we found that despite various roles each bucket plays in different types of attacks (due to the diversity in the content it serves), still the websites connecting to those buckets exhibit prominent common features (see Section 5.2.1), particularly, the presence of "gatekeeper" sites that cover the Bars (a valuable asset for the adversary) and remarkably homogeneous redirection behavior (i.e., fetching repository resources indirectly through other sites' references) and sometimes similar content organizations, due to the same attack payload the compromised sites upload from their backend (i.e., the Bars), or the templates the bucket provides to the adversary for quick deployment of her attack sites. By comparison, a legitimate bucket (e.g., reputable jQuery repository) tends to be *directly* accessed by the websites with highly diverse content.

Based on this observation, we developed *BarFinder*, a scanner that automatically detects Bars through inspecting the topological relations between websites and the cloud bucket they use, in an attempt to capture Bars based on the external features of the websites they serve. More specifically, for all the sites connecting to a repository, our approach correlates the domains and URLs (particular those related to cloud repositories) across their redirection chains and content features across their DOM structures to identify the presence of gatekeepers and evading behavior, and also measure the diversity of their content organization. A set of new *collective features*

generated in this way, including *bucket usage similarity*, *connection ratio*, *landing similarity* and others (Section 5.2.1), are further utilized by a classifier to find out suspicious buckets. Running the scanner over all the data collected by the Common Crawl [46], which indexed five billion web pages, for those associated with all major cloud storage providers (including Amazon S3, Cloudfront, Google Drive, etc.), we found around 1 million sites utilizing 6,885 repositories hosted on these clouds. Among them, BarFinder identified 694 malicious or compromised repositories, involving millions of files, with a precision of 95% and a coverage of 90% against our ground-truth set.

Our discoveries. Looking into the Bars identified by our scanner, we are surprised by the scope and the magnitude of the threat. These buckets are hosted by the most reputable cloud service providers. For example, 13.7% of Amazon S3 repositories and 5.5% of Google repositories that we inspected turned out to be either compromised or completely malicious[1]. Among those compromised are popular cloud repositories such as Groupon's official bucket. Altogether, 472 such legitimate repositories were considered to be contaminated, due to a misconfiguration flaw never reported before, which allows arbitrary content to be uploaded and existing data to be modified without proper authorization. The impact of these Bars is significant, infecting 1,306 legitimate websites, including Alexa top 300 sites like `groupon.com`, Alexa top 5,000 sites like `space.com`, etc. We reported our findings to Amazon and leading organizations affected by the infections. Groupon has already confirmed the compromise we discovered and awarded us for our help.

When it comes to malicious buckets, our study brings to light new insights into this new wave of repository based cyber-attacks, including the importance of Bars to malicious web activities and the challenges in defending against this new threat.

[1]We have manually examined and confirmed all those instances.

More specifically, we found that on average, one Bar serves 152 malicious or compromised sites. In one of the large campaigns discovered in our research, the Bar `cloudfront_file.enjin.com` hosts a malicious script that was injected into at least 1,020 websites (Section 5.3.1). These Bars sit right at the center of the attack infrastructure, supporting and coordinating other malicious actors' operations at different stages of a campaign. Interestingly, we found that they could be strategically placed on different cloud platforms, making them hard to block (due to the popularity of their hosting clouds like Google) and detect (scattered across different providers), and easy to share across multiple campaigns. As an example, the Potentially Unwanted Programs (PUP) campaign we found first loads a redirection script from a Bar on Akamaihd (the world's largest CDN platform) to lead the victim to the attack website, then fetches Phishing pictures from an Amazon S3 Bar, and finally delivers the malware stored on Cloudfront to the target systems (Section 6.4.3). In the presence of such meticulously planned attacks, the cloud service providers apparently are inadequately prepared, possibly due to the privacy constraints in touching their customers' repositories. We found that many *Bars* remain active during our study, and survive a much longer lifetime than that of the malicious content hosted on websites (Section 5.3.3). Further complicating the mission of Bar identification are other evasion techniques the adversary employs, including code obfuscation and use of a redirection chain and cloaking techniques to avoid exposing malicious payloads to a malware scanner.

Contributions. The contributions of the paper are as follows:

• *New understanding.* We performed the *first* systematic study on cloud repositories as a malicious service, an emerging security threat. For the first time, our study reveals the scope and magnitude of the threat and its significant impact, particularly on the infrastructures of illicit web activities. These findings bring to the spotlight this important yet understudied problem and lead to a better understanding of the

techniques the adversary employs and their weaknesses. This will contribute to better defense against and ultimate elimination of the threat.

• *New technique.* Based on our understanding of bad cloud repositories, we take a first step toward automatically detecting them. The technique we developed relies on the topological relationship between a cloud repository and the websites it serves, which are difficult to change and effective at capturing malicious or compromised buckets. Our evaluation over a large number of popular websites demonstrates the potential of the technique, which could be utilized by both cloud providers and third parties to identify the threats posed by *Bars*.

5.2 Finding Bars Online

In this section, we elaborate on our analysis of a set of known Bars (the seed set) and the features identified for differentiating benign repositories and Bars. These features are utilized in our research to build a simple web scanner, *BarFinder*, for detecting other malicious or compromised high-profile, previously-unknown repositories and the malicious campaigns in which they serve.

5.2.1 Features of Bad Repositories

Our study is based on a small set of confirmed good and bad repositories and their related domains, which we analyzed to find out how Bars (bad repositories) differ from legitimate repositories. In the absence of direct access to these buckets, good or bad, all we can do is to *infer* their legitimacy from who use them and how they are used (by different domains), that is, the features of the domains and their interactivities on the redirection paths leading to the cloud repository. Of particular interest here are a set of *collective* properties identified from the resource fetching chains (a.k.a., redirection chains) for serving the content of Bars, which is hard to change by the adversary, compared with the content features of individual Bars. Below, we elaborate on the way such data was collected and the salient features discovered in our research, which

describe how the adversary attempts to hide Bars or use them to cover other attack assets, a redirection pattern never observed on legitimate repositories.

Data collection. To build the seed set, we collected a set of confirmed malicious or compromised buckets (called *Badset*) and legitimate buckets (called *Goodset*) as well as their related domains, as illustrated in Table 14.

• *Badset*. We utilized two feeds as the ground truth for gathering bad cloud buckets: the *Spamtrap* feed and the *CleanMX* feed [42]. The former comes from a Spam honeypot we constructed [63] that receives around 10K Spam emails per day, from which cloud URLs promoted by the emails were extracted which may include spam resources such as HTML, images, and scripts. The latter includes the historical data of CleanMX, a popular domain scanning engine, from which cloud-related URLs were collected. For both feeds, we further validate them by VirusTotal [115] and manual inspections (e.g., looking for Phishing content) to ensure that they were indeed bad (to avoid contaminating the dataset with legitimate buckets used in malicious activities). Using the collected set of malicious cloud URLs from both feeds, we extracted their repositories, which led to 100 confirmed Bars.

• *Goodset*. The good buckets were gathered from the Alexa top 3K websites, which are considered to be mostly clean. To this end, we visited each website using a crawler (as a Firefox add-on) to record the HTTP traffic triggered by the visit, including network requests, responses, browser events, etc. From the collected traffic, we extracted the HTTP cloud request URLs corresponding to 300 cloud buckets hosted on 20 leading cloud hosting services like Amazon S3, Google Drive, etc. Note that even though some of them provide CDN service or DDOS protection, they are all provided hosting service to act as cloud repository.

• *Bucket-served sites and their HTTP traffic*. We collected HTTP traffic using the crawler mentioned above to visit a list of websites using buckets for feature extraction. Rather than blindly crawling the web to find those sites, we adopted a more targeted

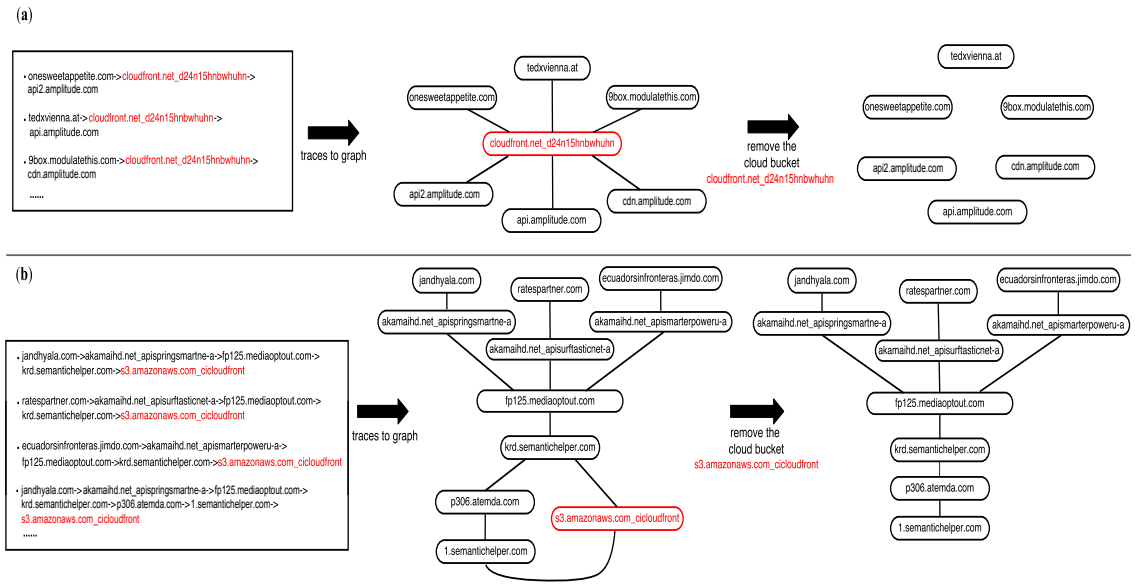


Figure 19: Example of the redirection infrastructure leading to the legitimate bucket *cloudfront.net_d24n15hnbwhuhn* **(a) and the Bar** *s3.amazonaws.com_cicloudfront* **(b), which are in RED color.**

strategy by crawling the sites found to contain links to the cloud in the past. We built the site list with the help of *Common Crawl* [46], a public big data project that crawls about 5 billion webpages each month through a large-scale Hadoop-based crawler and maintains lists of the crawled websites and their embedded links. Searching the *Common Crawl* [46] dataset, collected in February 2015, for the websites loading content from the 400 clean and malicious buckets identified above, we found 141,149 websites, were used by our crawler.

Topological features. We first inspected the *topology* of the redirection infrastructure associated with a specific bucket. Such an infrastructure is a collection of redirection paths, with each node being a *Fully Qualified Domain Name* (FQDN). On each path, the bucket is either a node when it directly participates in a redirection (e.g., its cloud URL delivers a redirection script to the visitor's browser) or simply a passive repository providing resources like pictures to other domains. Figure 19 illustrates examples of redirection paths leading to two real-world repositories, one for a legitimate bucket `cloudfront.net_d24n15hnbwhuhn` and the other for a Bar

Table 14: Summary results of the seed dataset.

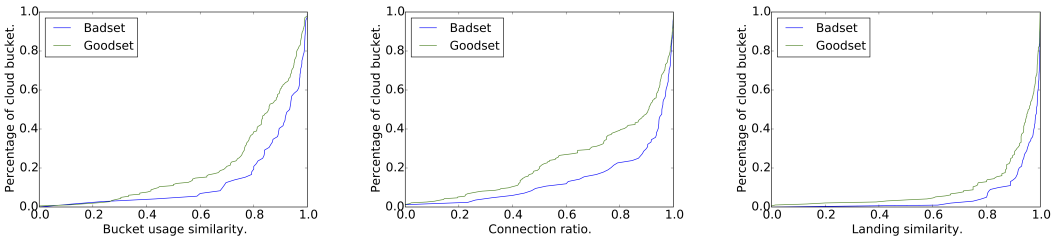
	# of buckets	# of linked websites	# of average linked website	# of redirection paths
Badset	100	12,468	133	468,480
Goodset	300	128,681	864	2,659,304

`s3.amazonaws.com_cicloudfront`.

A key observation from our study is that *the redirection infrastructure leading to a Bar tends to include the features for protecting the Bar from being detected by web scanners*, presumably due to the fact that the repository is often considered to be a valuable asset for the adversary. Specifically, we found that typically, there are a few *gatekeeper* nodes sitting in front of a Bar, serving as an intermediary to proxy the attempts to get resources from the Bar. Examples of the gatekeepers include `fp125.mediaoptout.com` and its downstream nodes in Figure 19(b). On the topology of such an infrastructure, these gatekeepers are the hubs receiving a lot of resource-access connections from entry sites (the first node on a redirection path, see Figure 19). Also interestingly, our research shows that some gatekeepers can access the Bar through multiple paths. For example, in Figure 19(b), `krd.semantichelper.com` can either go straight to `s3.amazonaws.com_cicloudfront` or take a detour through `p306.atemada.com`. This structure could be caused by the cloaking of the gatekeeper for hiding the Bar, or constructed to maintain access to the repository even when nodes (like `1.semantichelper.com`) are down (detected, cleaned, etc.). Note that such a protection structure does not exist on the paths to a benign repository (Figure 19(a)): normally, the resources hosted in a repository (e.g., jQuery) is directly fetched by the website using it, without going through any redirection; even in the presence of redirections, there will not be any gatekeeper, not to mention attempts to cloak or build a backup path.

To identify this unique "protection" structure, we utilize two *collective features*: *bucket usage similarity* (BUS) that captures the topology involving hubs (gatekeepers) and *connection ratio* (CR) that measures the interactivities across different redirection paths (which point to the existence of cloaking behavior or the attempts to maintain back-up paths to the Bar). Specifically, consider a redirection graph $G = (V, E)$ (as illustrated in Figure 19), where V is the set of nodes (the FQDNs involved in a redirection) and E is a set of edges from one node to the next one on individual paths: $E = \{e_{i,j} | node\ i\ precedes\ node\ j\ on\ a\ path\}$. The BUS is measured by $1 - \frac{i}{s}$, where i is the number of immediate predecessor nodes to a repository (the domains connecting to the repository) and s is the total number of entries of the repository's redirection graph. To find out the CR, we first remove the bucket b and all the edges to which it is attached (if they exist) to get another graph $G' = G - G_b$, where $G_b = (\{b\}, E_b)$ and $E_b = \{e_{b,j}\}$. Note that each graph G' is associated with one bucket. Then, from G', we find out the number of connected components n and calculate $CR = 1 - \frac{n}{|V|}$ (see Figure 19 for an example).

Both collective features were found to be discriminative in our research. Figure 20(a) and 20(b) compare the cumulative distributions (CDF) of the ratios between Bad and Good sets. As we can see from the figures, Bars tend to have higher ratios than benign ones: the average BUS is 0.87 for the Bars and 0.79 for the legitimate repositories and the CR is 0.85 for the bad repositories and 0.67 for the good one. As mentioned earlier, this is caused by the fact that a small set of gatekeepers nodes are often placed there for protecting the Bars while the redirection chains towards the good repositories are much more direct and independent: different organizations typically do not go through an intermediary to indirectly access the public repository like jQuery, and even within the same organization, use of such a resource is often direct. Although there can be exceptions, our measurement study shows that in general, the structural differences between malicious and legitimate repositories are stark.



(a) **Cumulative distribution of bucket usage similarity per cloud bucket.**

(b) **Cumulative distribution of connected ratio per cloud bucket.**

(c) **Cumulative distribution of landing similarity per cloud bucket.**

Figure 20: Bars show smaller topological diversity.

Also, we found that occasionally, a Bar itself may serve as a gatekeeper, running scripts to hide more valuable attack assets, such as the attack server or other malicious landing sites. When this happens, almost always the Bar leads to a small set of successors on redirection paths (e.g., attack servers, land sites). This is very different from the redirection performed by the script from a benign repository, for example, `cloudfront.net_d24n15hnbwhuhn`. In such cases, the targets of redirections are often very diverse. Based on this observation, we further measure the *landing similarity*, $LS = 1 - \frac{l}{s}$, where l is the number of the unique last nodes on the redirection paths associated with a repository. Again, as illustrated in Figure 20(c), our study shows that redirection paths involving Bars share fewer end nodes than legitimate ones, and therefore, the related redirection graphs (for Bars) have a higher landing similarity (0.94 vs 0.88).

Content and network features. In addition to their distinctive topological features, we found that the nodes on the redirection paths attached to a Bar often exhibit remarkable homogeneity in their content and network properties. Particularly, for the websites directly connecting to the repository, we found that they typically use a small set of templates (like WordPress) to build up their web pages, include similar DOM positions for script injection, carrying similar IP addresses or even having the same content management system (CMS) vulnerabilities, etc. These properties turn out to

be very diverse among those utilizing a legitimate cloud repository. For example, all websites linking to a Google Drive Bar have their malicious cloud URL (for injecting a script) placed at the bottom of the DOM of each website. In another example, we found that the front-end sites using a Cloudfront Bar actually all include a vulnerable JCE Joomla extension.

To better understand the diversity of such websites, we try to compare them according to a set of content and network properties. In our research, we utilized the properties extracted by *WhatWeb* [120], a popular webpage scanner. WhatWeb is designed to identify the web technologies deployed, including those related to web content and communication: e.g., CMS, blogging platforms, statistic/analytics packages, JavaScript libraries, social media plugins, etc. For example, from the content

```
<link rel="search"
  type="application/opensearchdescription+xml"
  href="https://wordpress.com/opensearch.xml"
  title="WordPress.com" />
```

we obtain the property p as a key-value pair $p = (k, v) = (wordpress, opensearch)$, which indicates the website using wordpress plugin opensearch.

From our seed dataset, the scanner automatically extracted 372 keys of 1,596,379 properties, and then we clustered the keys into 15 classes such as Analytics and tracking, CMS and plugin, Meta-data information, etc., following the categories used by *BuiltWith*, a web technology search engine [36]. Some examples of these properties are presented in Table 15. In addition to these properties extracted by *WhatWeb*, we added the following properties to characterize cloud URLs, including the position of the URL, the order in which different buckets appear in the web content and the number of cloud platforms used in a page.

Based on these properties, again we utilized a topological metric to measure the

Table 15: Examples of content and network features.

Category	Feature	Example
Content	CMS platform information and their plugin	(wordpress, all in one SEO pack)
	Meta-data information	(metagenerator, drupal7)
	CloudURL information	(position, bottom)
	Advertising	(adsense, asynchronous)
	Javascript library	(JQuery, 1.9.1)
	Analytics and tracking	(Google-Analytics, UA-2410076-31)
	Widget	(addthis, welcome bar)
	DocInfo technologies	(open graph protocol, null)
Network	Identity	(IP, 216.58.216.78)
	Cookie	(Cookie, harbor.session)
	Server framework version	(Apache, 2.4.12)
	Custom HTTP header	(X-hacker, If youre..)

overall similarity across sites. Specifically, the relations among all the sites (connecting to the same bucket) in the same category (Analytics and tracking, CMS and plugin, etc.) are modeled as a graph $G' = (V', E', P)$, where V' is the set of the websites, which are characterized by a collection of properties P, and E' is the set of edges: $E' = \{e_{i,j} | website\ i\ and\ j\ share\ p \in P\}$, that is, both sites having a common property. Over this graph, the *site similarity* is calculated as $SiS = 1 - \frac{n}{|V'|}$. Here n is the number of connected components in the graph.

In our research, we computed SiS across all the categories summarized from the seed dataset, and compared those with Bars against those with the legitimate buckets.

Again, the sites using Bars are found to share many more properties and therefore achieve a much higher similarity value than those linking to a good bucket. This is likely caused by mass-production of malicious sites using the same resources (templates, pictures, etc.) provided by a Bar or utilization of the same exploit tool stored in a Bar for compromising the sites with the same vulnerabilities. Therefore, such similarity is inherent to the attack strategies and can be hard to change.

5.2.2 BarFinder

Design. The design of BarFinder includes a web crawler, a feature analyzer, and a detector. The crawler automatically scans the web for cloud buckets (embedded in web content) and then clusters websites according to the buckets they use. From each cluster, the analyzer constructs a redirection graph and a content graph as described earlier (Section 5.2.1), on which it further calculates the values for a set of collective features including disconnection ratio (D), bucket usage similarity (B), landing similarity (L) and a series of content property/network property similarities ($S_1 \cdots S_n$) for n web-technology categories (e.g., analytics and tracking, CMS and plugin, meta-data information, etc.). The output of this feature analysis is then passed to the detector, which maintains a model (trained on the seed dataset) to determine whether a bucket is malicious, based on its collective features.

Specifically, the crawler visits each website, inspecting its content, triggering events, recording the redirection paths it observes and parsing URLs encountered using the patterns of known cloud platforms to recognize cloud buckets. For example, the repository on Amazon S3 is accessed through the URL formatted as $w + .s3\{-w+\}[?].amazonaws.com$, and Amazon CloudFront produces resource URLs in the form of $w + .cloudfront.net$. In our research, 20 cloud platforms were examined to identify the buckets they host. At the feature-analysis stage, for each bucket, BarFinder inspects all its redirection paths, converts every node into an FQDN to

compute their topological features, and then connects different nodes according to their content and network properties to find out their site similarities, as described in Section 5.2.1.

Next, each cloud bucket i is uniquely characterized by a vector: $\langle D_i, B_i, L_i, S_{i,1} \cdots S_{i,n} \rangle$, with each element a collective feature. Individual features have different power in differentiating good and bad buckets, which we measured using the F-Score [33] (see Table 16). Note that the feature with a large score can better classify these vectors than the one with a small value. Therefore, a binary classifier with a model for weighing the features and other parameters can be used to classify the vector set and determine whether individual buckets are legitimate or not. Such a model is learned from the seed dataset. In our research, we utilized a Support Vector Machine (SVM) as the classifier, which showed the best performance among other classification algorithms (see Table 17). Its classification model is built upon the F-Scores for the collective features (D, B, etc.) and a threshold set according to the false positive and negative discovery expected to achieve. For each bucket classified, the SVM can also report the confidence of the classification.

Implementation. This simple design was implemented in our study into a prototype system. The web crawler was built as a Firefox add-on. In total, 20 such crawlers were deployed. We further developed a tool in Python to recover cloud URLs from the web content gathered by Common Crawl. The feature analyzer includes around 500 lines of Python code for processing the data collected by the crawler and computing the collective features (Section 5.2.1). Each feature in the vector is normalized using the L1 norm before passed to the SVM classifier. In our system, we incorporated the SVM provided by the scikit-learn open-source machine learning library [106].

Table 16: F-score of features.

Feature	Label	Metric	F-score
Connection ratio	D	$1 - \frac{n}{\|V\|}$	0.084
Bucket usage similarity	B	$1 - \frac{i}{s}$	0.076
Landing similarity	L	$1 - \frac{l}{s}$	0.072
CMS information	S_1	$1 - \frac{n}{\|V'\|}$	0.037
Meta-data information	S_2	$1 - \frac{n}{\|V'\|}$	0.033
Analytics and tracking	S_3	$1 - \frac{n}{\|V'\|}$	0.032
Widget	S_4	$1 - \frac{n}{\|V'\|}$	0.031
CloudURL information	S_5	$1 - \frac{n}{\|V'\|}$	0.024

5.2.3 Evaluation

Here we report our evaluation of BarFinder on both the ground truth and the Unknown sets. All the experiments were conducted within an Amazon EC2 C4.8xlarge instance equipped with Intel Xeon E5-2666 36 vCPU and 60GiB of memory.

Evaluation on the seed set. We tested the effectiveness of BarFinder over our ground-truth dataset (i.e., the seed set) through the standard five-fold cross validation: that is, 4/5 of the data was used for training the SVM and the remaining 1/5 for evaluating the accuracy of Bar detection. Specifically, we randomly chose 80 Bars (out of 100) from the Badset and 240 (out of 300) legitimate buckets from the Goodset, together with the related websites (out of 141,149). These data were first processed by our prototype to adjust the weights and other parameters for its model. Then we tested the model on the remaining dataset (20 Bars, 60 legitimate buckets). The process is then repeated 5 times. BarFinder achieved both a low false discovery rate (FDR: 1- precision) and a high recall in detection: only 5.6% of reported Bars turned out to be legitimate (i.e., 1.6% of false positive rate), and over 89.3% of the Bars were detected. We further show the Area Under Curve (AUC) of the Receiver Operating Characteristics (ROC) graph, which comes very close to 1 (0.96), demonstrating the good balance we strike between the FD rate and the coverage. This preliminary analysis shows that the collective features of the sites connecting to

Table 17: Performance comparison under five-fold cross validation.

Classifier	Precision	Recall
SVM	0.94	0.89
Decision Tree	0.9	0.83
Logistic Regression	0.91	0.87
Naive Bayes	0.9	0.79
Random Forest	0.85	0.82

cloud repositories are promising in detecting Bars.

Evaluation on the Unknown set. We now use BarFinder to scan an unknown set. This unknown set contains HTTP traffic collected using a crawler as described in Section 5.2.1 to visit a list of websites. This list of websites is also extracted from common crawl [46] by searching for websites that have loaded some content in the past from the cloud platforms. As a result, the unknown data set contained HTTP traffic generated from dynamically visiting 1M websites loading content from 20 cloud platforms and 6,885 cloud buckets.

To validate our evaluation results, we employ a methodology that combines anti-virus (AV) scanning, blacklist checking, and manual analysis. Specifically, for the Bars flagged by our system, we first scan their cloud URLs with VirusTotal for malware and check them against the list of suspicious cloud URLs collected from our Spamtrap honeypot for Spam, Phishing, blackhat Search Engine Optimization (SEO), etc. In the case of VirusTotal, a URL is considered to be suspicious if at least two scanners raise the alarm. All such suspicious URLs (from either VirusTotal or the Spamtrap list) are cross-checked against the blacklist of CleanMX. Only those also found there are reported to be a true positive. Once a URL is confirmed malicious, its corresponding bucket is labeled as bad. Those unlabeled but flagged (by BarFinder) buckets are further validated manually.

In the experiment, BarFinder reported a total of 730 Bars, about 10.6% of the 6,885 buckets. Among them, the AV scanning and blacklist verification confirmed that 502 buckets were indeed bad. The remaining 228 were manually analyzed through,

e.g., inspecting the resources in the buckets for phishing or scam content, running scripts in the VM to capture binary code download. This validation further confirmed 192 Bars. The FDR was found to be at most 5% (assuming those not confirmed to be legitimate), in line with the finding from the seed set.

5.3 Measurement and Discoveries

Based on the discoveries made by BarFinder, we further conducted a measurement study to better understand the fundamental issues about Bar-based malicious services, particularly *how the cloud repositories help facilitate malicious activities*, *how the adversary exploited legitimate cloud buckets* and *why the adversary uses Bars in the first place*. Our research shows that on the infrastructure, Bars play a pivotal role, compared with the content kept on other malicious or compromised sites, possibly because they are hosted on popular cloud services, and therefore hard to blacklist and also easy to share across different campaigns. Also, in a malicious campaign, the adversary may take advantage of multiple Bars, at different attack stages, to construct a complicated infrastructure that supports her mission (Section 5.3.1). More importantly, we discovered that the adversary effectively exploited misconfigured legitimate buckets to infect a large number of their front-end web services (Section 5.3.2), and the cloud providers have not done much to counteract the threat, often leaving Bars there for a long time (Section 5.3.3), possibly due to the privacy constraints and limited means to detect individual components of a malicious activity. Such observations, together with the challenge in blocking Bars, offer insights into the motivation for moving toward this new trend of repository-based attacks.

5.3.1 Bar-based Malicious Web Infrastructure

Landscape. As mentioned earlier, BarFinder reported 730 suspicious repositories from 6885 cloud buckets over 20 cloud platforms. Among them, we utilized 694 confirmed Bars (through AV/blacklist scanning or manual validation, see Section 5.2.3)

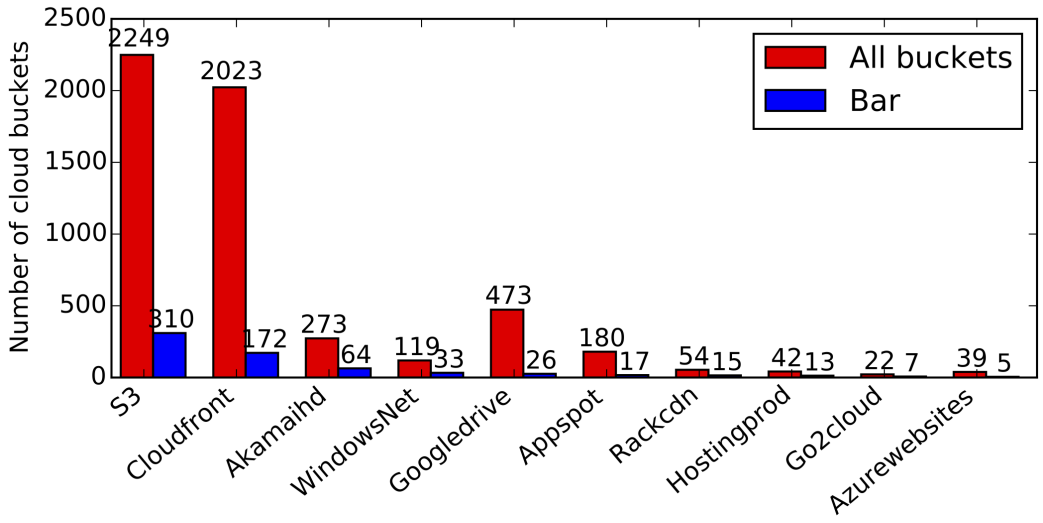


Figure 21: Top 10 cloud platforms with most Bars, compared with their total number of cloud buckets in our dataset.

for the measurement study. These Bars were found to directly serve 156,608 domains (i.e., front-end websites), through which they are further attached to 6,513,519 redirection paths involving 166,772 domains. Figure 21 illustrates the number of Bars we found on different cloud platforms. Among them, Amazon S3 is the most popular one in our dataset, hosting the most Bars (45%), which is followed by CloudFront (Amazon's CDN) 25.1% and Akamaihd 9.3%. Note that of these 20 clouds, seven of them provide free storage services (e.g., 15GB free space on Google Drive, 5GB for Amazon S3), and therefore easily become the ideal platforms for low-budget miscreants to distribute their illicit content. Also, eleven of them support HTTPS, on which malicious activities are difficult to catch by existing signature-based intrusion detection systems like snort and Shadow[107][113]. Interestingly, on some of the most prominent platforms, the miscreants are found to take advantage of the cloud providers' reputations to make their Phishing campaigns look more credible: for example, we found that the adversary continuously spoofed Gmail's login page on Google Drive, and the software download page for Amazon FireTV in an Amazon S3 bucket.

Figure 33 shows the distribution of Bars' frontend websites across 81 countries,

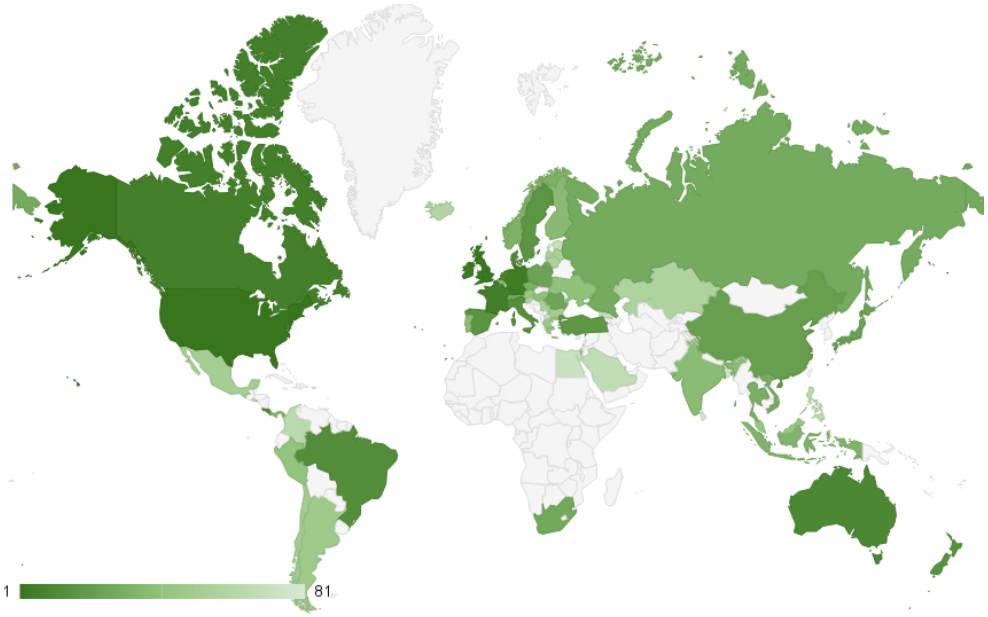


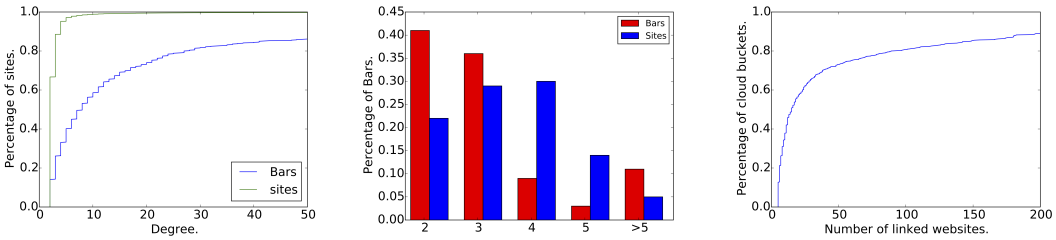
Figure 22: Impact of Bars' front-end websites around the globe.

Table 18: Top 10 most popular Bars.

Rank	Cloud bucket	# of front-end sites	Avg path len	Popularity
1	s3.amazonaws.com_content.sitezoogle.com	4,429	2.9	2.8%
2	cloudfront.net_d3n8a8pro7vhmx	1,829	3.3	1.4%
3	s3.amazonaws.com_assets.ngin.com	1,643	3.2	1.2%
4	s3.amazonaws.com_publisher_configurations.shareaholic	1,434	2.7	0.9%
5	cloudfront.net_d2e48ltfsb5exy	1,340	4.0	0.9%
6	cloudfront.net_d1t3gia0in9tdj	1,297	3.2	0.9%
7	cloudfront.net_d2i2wahzwrm1n5	1,249	2.5	0.8%
8	cloudfront.net_d202m5krfqbpi5	1,062	2.8	0.8%
9	s3.amazonaws.com_files.enjin.com	1,020	7.1	0.7%
10	akamaihd.net_cdncache3-a	976	6.4	0.6%

as determined by the geolocations of the sites. The number of Bars' frontend sites in each country is ranked and described with different levels of darkness in the figure. We observe that most of these frontends stay in United States (14%), followed by Germany (7%) and United Kingdom (5%).

Role in attack infrastructures. Actually, most nodes on a malicious infrastructure are the malicious websites with newly registered domains and those that are compromised. To better understand the critical roles of Bars, we compared those nodes with the bad cloud buckets. Specifically, we first identified both types of nodes from the redirection paths and then analyzed the number of unique paths each member in



(a) **Cumulative distribution of degrees per sites.**

(b) **Percentage of Bars in each position of redirection path (Ignoring those traces with length of 2).**

(c) **Cumulative distribution of number of in-degrees per Bar.**

Figure 23: Bars play critical roles in attack infrastructures.

either category is associated with and the position of the member on the path. Figure 23(a) presents the cumulative distribution of the paths going through a Bar and that of a compromised or malicious site. As seen in the figure, compared with other nodes on the infrastructure, Bars clearly sit on much more paths (47.4 on average vs. 8.6), indicating their importance.

Further, Figure 23(b) shows the histogram of position distributions (again, Bars vs. bad sites). The observation is that more Bars (41%, 11%) show up at the beginnings and the ends of the paths than bad websites (22%, 5%), which demonstrates that they often act as first-hop redirectors or attack-payload repositories. For example, in our three-month-long monitoring of the campaign based on the Spyware distribution Bar `akamaihd.net_rvar-a`, we found that besides the Bar, 320 newly-registered websites participated in the attack; here the Bar acted very much like a dispatcher: providing JavaScript that identified the victim's geolocation and then using an iframe to redirect her to a selected bad site.

Content sharing. Our research reveals that Bars have been extensively shared among malicious or compromised websites, also across different positions on malicious redirection chains. Figure 23(c) illustrates the cumulative distribution of Bars' in-degrees in their individual redirection graphs: that is, the number of the sites utilizing these Bars. On average, each Bar shows up on 252 sites and 12% of them are

used by more than 200 websites. Table 18 lists the 10 most popular Bars we found. Among them, eight, including `s3.amazonaws.com_content.sitezoogle.com`, `s3.amazonaws.com_publisher_configurations.shareaholic`, etc., host services for website generation, blackhat SEO or Spam. Particularly, `akamaihd.net_cdncache3-a` turns out to be a distributor of Adware, whose scripts are loaded into the victim's browser to redirect it to other sites for downloading different Adware. Also, we found that another Bar `s3.amazonaws.com_files.enjin.com` hosts exploits utilized by 1,020 bad sites. Finding Bars can help to effectively detect more sites with malicious contents.

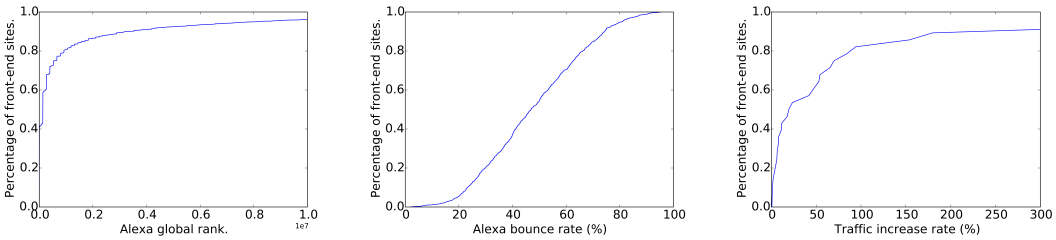
Another interesting observation is that malicious content is also extensively shared across different Bars. To understand such content reuse, we grouped the malicious programs retrieved from different Bars based on the similarity of their code in terms of edit distance. Specifically, we removed the spaces within the programs and ran the Python library *scipy.cluster.hierarchy.linkage* [103] to hierarchically cluster them (now in the form of strings) according to their Jaro scores [43]. In this way, we were able to discover three types of content sharing: intra-bucket sharing, cross-bucket sharing, and cross-platform sharing. Specifically, within the Amazon bucket `akamaihd.net_asrv-a`, we found that many of its cloud URLs are in the form of `http://asrv-a.akamaihd.net/sd/[num]/[num].js`. The JavaScript code turns out to be all identical, except that each script redirects the visitor to a different website. The similar code also appears in another Amazon bucket `akamaihd.net_cdncache-a`. As another example, we discovered the same malicious JavaScript (`JS.ExploitBlacole.zm`) from the Bars on CloudFront and Qiniudn respectively, even under the same path (i.e., `media/system/js/modal.js`). Moreover, we found that attackers used sub-domain generation algorithm to automatically generate sub-domain for Bars, then further reused the same malicious contents for these Bars.

Specifically, we found that 28 content sharing Bars on Akamaihd have the same format in their names. Attackers utilized a word bank based sub-domain generation algorithms [48], which concatenates fixed terms and a series of domain names (remove dot), then truncates the string if its length is over 13, e.g., `apismarterpoweru-a` (truncated from `smarterpowerunite.com`). The common patterns of Bars indicate the potential of developing an accurate detection procedure.

Correlation. We further studied the relationships between different Bars, fetched by the same websites. From our dataset, 11,442 (3.5%) websites are found to access at least two Bars. Among them, 8,283 were served as front-end websites, and 3,159 other sites on redirection chains. Also, 60.9% of these sites link to the repositories on the same cloud platforms and 39.1% use those on different platforms. In some cases, two buckets are often used together. For example, we found that a click-hijacking program was separated into the code part and the configuration part: the former is kept on CloudFront while the latter is on Akamaihd; the two buckets always show up together on redirection chains. Such a separation seems to be done deliberately, in an attempt to evade detection. Also we saw that Bars carrying the same attack vectors are often used together, which are apparently deliberately put there to serve parties of the same interests: as another example, a compromised website was observed to access four different Bars on different cloud platforms, redirecting its visitors to different places for downloading Adware to the visitor's system. Our findings show that Bars are widely deployed in attacks and serve in a complex infrastructure.

5.3.2 Bucket Pollution

Polluted repositories. To find polluted buckets, we searched the Alexa top 20K websites for the Bars in our dataset and 276 Bars were found. When a legitimate site links to a Bar, the reason might be either the website or the repository is hacked. Differentiating these two situations with certainty is hard, and in some cases, it may



(a) **Cumulative distribution of Alexa global ranks per Bars' front-end sites.**

(b) **Cumulative distribution of Alexa bounce rate per Bar's front-end sites.**

(c) **Cumulative distribution of traffic increase rate per Bar's front-end sites.**

Figure 24: Alexa global rank, bounce rate and traffic increase rate of Bar's front-end websites.

not be possible. All we could do is to get an idea about the prevalence of such bucket pollution, based on the intuition that if a website is less vulnerable, then it is less likely to be compromised. To this end, we ran WhatWeb, a powerful web vulnerability scanner, on these sites and found 134 Bar's front-end websites contain various flaws, such as using CMS in vulnerable version (e.g. wordpress 3.9), vulnerable plugins (e.g., JCE Extension 2.0.10) and vulnerable software (e.g., Apache 2.2). The remaining 142 Bar's front-end websites look pretty solid in web protection and therefore it is likely that the Bars they include were polluted. This set of potentially compromised buckets takes 19% of all the Bars flagged by BarFinder. These buckets, together with the additional 30 randomly sampled from the set, went through a manual analysis, which shows that indeed they were legitimate buckets contaminated with malicious content.

Misconfiguration and impact. It is even more challenging to determine how these buckets were compromised, which could be caused by exploiting either the cloud platform vulnerabilities or the bucket misconfigurations. Without an extensive test on the cloud platform and the repositories, which requires at least direct access to them, a comprehensive study on the issue is impossible. Nevertheless, we were able to identify a misconfiguration problem widely existing in popular buckets. *This flaw has never been reported before but was likely known to the underground community and has already been utilized to exploit these repositories. We reported the flaws to*

```
GET /?delimiter=/ HTTP/1.1
Host: (bucket-name).s3.amazonaws.com
Accept-Encoding: identity
content-length: 0
Authorization: AWS (access key):(secret key)
```

Figure 25: Constructed request header.

the vendors and they confirmed our finding.

Specifically, on Amazon S3, one can configure the access policies for her bucket to defines which AWS accounts or groups are granted access and the type of access (i.e., list, upload/modify, delete and download): this can be done through specifying access control list on the AWS Management Console. Once this happens, the cloud verifies the content of the `authorization` field within the client's HTTP request header before the requested access is allowed to go through. However, we found that by default, the policy is not in place, and in this case, the cloud only checks *whether the authorization key (i.e., access key and secret key) belongs to an S3 user, not the authorized party for this specific bucket*: in other words, anyone, as long as she is a legitimate user of the S3, has the right to upload/modify, delete and list the resources in the bucket and download the content. Note that this does not mean that the bucket can be directly touched through the browser, since it does not put anything into the authorization field. However, the adversary can easily build his own HTTP header, filling in his own S3 key, as illustrated in Figure 25, to gain access to the misconfigured repository. In our research, we verified that all such operations can be performed on any repositories with the configuration flaw, which suggests that site operators need to take more caution when setting the configuration rules.

To understand the impact of this problem, we developed a simple web testing tool, which checked a bucket's configuration using our own S3 key. By scanning all 6,885 repositories (including both Bars and legitimate buckets), we discovered that 472 are vulnerable, which were associated with 1,306 front-end websites. The Alexa global

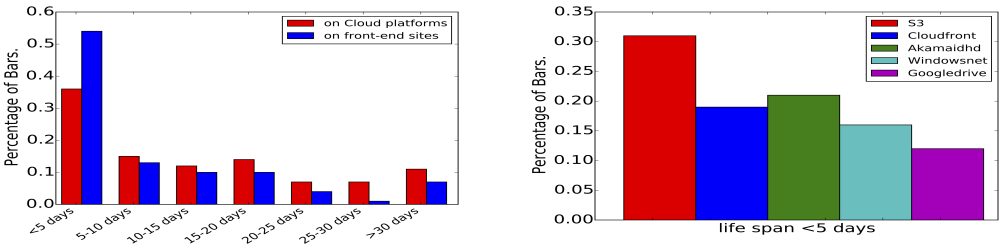
ranks and the bounce rates of their front-end websites are illustrated in Figure 35(a) and Figure 24(b). 63% of them have bounce rates from 20% to 60%; 9 sites are ranked within Alexa top 5000 (e.g., `groupon.com`, `space.com`).

Focusing on the 104 bad buckets with the flaws, we further manually sampled 50 and confirmed that these buckets were indeed legitimate, including high-profile ones like `s3.amazonaws.com_groupon`. Further, looking into the these buckets' file uploading time (retrieved from the buckets through the flaw), we found that in some cases, the attack has been there for six years. Particularly the Amazon bucket `s3.amazonaws.com_groupon`, Groupon's official bucket, was apparently compromised five times between 2012 and 2015 (see Section 6.4.3 for details), according to the changes to the bucket we observed from the bucket historical dataset we collected from `archive.org`. We also estimated the volume of traffic to those Bar-related sites using a PassiveDNS dataset [51], which contains DNS lookups recorded by the Security Information Exchange. Figure 24(c) illustrates the traffic of the websites during the time period when their buckets were compromised, which was increased significantly compared with what those sites received before their compromise, indicating that they likely received a lot of visits. This provides evidence that the impact of such compromised buckets is indeed significant.

5.3.3 Lifetime and Evasion

In the presence of the severe threat from Bars, we found that cloud providers' responses, however, are far from adequate. This is highlighted by the relatively long lifetimes of malicious repositories we observed.

Lifetime. To understand the duration of Bars' impacts, we continuously crawled the front-end bad sites every five days to check whether they were still using the same set of Bars, and also malicious cloud URLs to find out whether the repositories were still alive. Figure 35(c) illustrates the distributions of such bad repositories' life



(a) **Distributions of Bars' life spans on front-end sites and on cloud platforms.** (b) **Percentage of Bars removed within 5 days in top 5 cloud platforms with most Bars.**

Figure 26: Lifetime of Bars.

spans within those front-end sites and on cloud platforms. As can be seen in the figure, on average, the references of these Bars on the websites were removed much faster than their cloud URLs and ultimately their accounts on the cloud platforms. Apparently, the cloud providers are less aggressive, relative to the website owners, in addressing Bar-related infections. In Figure 26(b), we further compare Bars' life spans on different platforms: interestingly, with more bad buckets on its servers, Amazon AWS acted more promptly than other clouds; Google, however, moved much slower: for example, on Google Drive, a repository hosting malware-serving pages, `googledrive.com_0B8D1eUrPT_z3OVpBTVJ3LUg2UEk`, stayed there for over 150 days, longer than the average duration of other exploit servers (non-cloud) reported by the prior work [60][79] (2.5 hours). The observation indicates that cloud providers have noticed such problem, but a likely lack of effective methods to identify and clean Bars.

Evasion. Such a long lifetime could be related to a spectrum of evading techniques the adversary deploys to protect his cloud assets, which are described as follows:

• *Content separation.* Apparently, the adversary tends to break his attack assets into pieces and store them at different places. As mentioned earlier, we found that malware's code and configuration files were placed in different buckets. Also, we discovered in this study that there are 32 Bars that host nothing but images used in various attacks, Phishing and Fake AV campaigns in particular. Since the images

themselves do not contain any malicious code, these repositories typically stay on the clouds for a long time, >30 days on average.

• *Content change.* Another interesting observation is that the malicious content within Bars changes over time, in an attempt to avoid being linked to blacklisted malicious websites. Specifically, looking into the history of the content (from *archive.org*) retrieved from the Bar through the same cloud URL, we found that part of the content (e.g., the destination of a redirection) changes frequently, moving quickly away from known malicious sites.

• *Redirect cloaking.* Like malicious or compromised websites, Bars are also found to leverage cloaking techniques (rendering different content based on the visitor' cookie, IP, user agent, etc.) to avoid detection. However, different from websites, cloud hosting services typically do not support server-side scripting. As a result, Bars have to run the cloaking code on the client (browser) side, which makes the evasion less stealthy. To make up for this weakness, the adversary was observed to place redirection websites in front of Bars, running cloaking techniques there to hide the repositories.

• *Obfuscation.* We found that the attack payloads in the repositories were often obfuscated. Various kinds of obfuscation techniques were found from simply Base64 encoding to online obfuscation tools (e.g., `api.myobfuscate.com`). Actually, even the links to refer to Bars within front-end websites were obfuscated in some cases, apparently, for the purpose of protecting the repositories.

Further, our study shows that these techniques were also utilized together to make identification of Bars even harder. Specifically, we manually choose 10 Bars with each evasion technique (40 in total), combined with 10 Bars without evasion technique, and then compare their life spans. It is clear that evasion techniques do allow Bars to hide longer, as illustrated in Table 19.

Table 19: Comparison of Bars' lifetime under different evasion techniques.

Evasion technique	# of Bars	# of front-end sites	Avg. life span
Content separation	10	743	25-30 days
Content change	10	1045	>30 days
Redirect cloaking	10	1220	10-15 days
Obfuscation	10	1032	10-15 days
None	10	984	5-10 days

5.3.4 Case Studies

In this section, we discuss two prominent examples.

PUP campaign. Our study reveals a malicious web campaign dubbed *Potentially Unwanted Programs* (PUP) distribution: the attack redirects the victim to an attack page, which shows her fake system diagnosis results or patch requirements through the images fetched from a Bar, in an attempt to cheat the victim into downloading "unwanted programs" such as Spyware, Adware or a virus. This campaign was first discovered in our dataset. Altogether, at least 11 Bars from 3 different cloud platforms and 772 websites (not hosted on the cloud) were involved in.

Through analyzing the redirection traces of the campaign, we found that two Akamai Bars, `akamaid.net_cdncache3-a` and `akamaihd_asrv-a`, frequently inject scripts into compromised websites, which serve as first-hop redirectors to move a visitor down the redirection chain before hitting malicious landing pages (that serve malicious content). Interestingly, all the follow-up redirectors are compromised or malicious websites that are not hosted on the cloud. The scripts in the Bars were found to change over time, redirecting the visitor to different next-hop sites (also redirectors). On average, the life span of such sites is only 120 hours, but the Bar was still alive when we submitted this paper. Such redirections end at at least 216

malicious landing sites, which all retrieve deceptive images from an Amazon S3 bucket `s3.amazonaws.com_cicloudfront` (a Bar never reported before and is still alive). An example is a system update warning, as shown in Figure 18. From the repository, we collected 134 images, including those for free software installation, updates on all mainstream OSes, browsers and some popular applications. If she clicks and downloads the program promoted on the site, the code will be fetched from multiple Bars, such as `s3.amazonaws.com_wbt_media` where the PUP puts a Bitcoin miner on the victim's system, and `cloudfront.net_d12mrm7igk59vq`, whose program modifies Chrome's security setting.

Groupon Bar. We discovered that a misconfigured Amazon S3 bucket `s3.amazonaws.com_groupon` belongs to Groupon (Alexa global rank 265), a global e-commerce marketplace serving 48.1 million customers worldwide. The bucket was used as the resource repository for Groupon's official website (i.e., `groupon.com`) as well as its marketing sites (12 websites observed in our dataset). When tracking its historical content from `archive.org`, we were surprised to see that the Groupon S3 bucket has been compromised at least eight times in the past five years (e.g., 2015/08/06, 2014/12/18, 2014/06/25, 2014/01/27, 2014/02/26, 2013/06/23, 2011/11/08, 2010/09/28). These attacks caused different types of malicious payloads to be uploaded to their repository, including Adware, Trojan, virus and others. Even though the bucket owner changed the access control policy in 2012 to prevent the unauthorized party from directly listing the bucket content through browser, it remained accessible by our tool mentioned in Section 5.3.2, which constructs an *Authorization* field in HTTP header, and unauthorized listing, upload and even modification can still occur.

5.4 Discussion

Limitations of BarFinder. As mentioned earlier, Bar detection is hard, since cloud repositories cannot be directly accessed by the parties outside the cloud. Therefore,

the goal of BarFinder is to leverage the sites served by Bars to find suspicious repositories. For this purpose, we chose to utilize the *collective* features of these sites, such as their topological relations, content shared across sites, etc. This strategy could make the approach more robust, as the collective features are more difficult to evade compared with those from individual sites. On the other hand, it requires that the party running the system first makes efforts to gather the sites using cloud buckets, the more the better. Further, there are repositories that only serve a small set of front-end sites: e.g., we found that among the Alexa top 3K sites, 67 sites are connecting to the cloud buckets only used by themselves. Those "self-serving" buckets are rather popular in reputable websites such as `appspot.com_android-site` only used by `android.com`, `s3.amazonaws.com_ttv-backgroundart` only used by `twitch.tv`, etc. This fact makes the bad apples among them hard to catch by BarFinder simply because not enough sites using them are out there to allow us to differentiate these two types of repositories. Detection techniques covering this type of Bars need to be developed in the follow-up research.

Other defenses against Bars. Besides the detection effort made by the third party, as BarFinder does, more can be done to mitigate the threats posed by Bars, from the ends of the website owner, the bucket owner and the service provider. The website owner could perform integrity checks on the resources her website retrieves from the bucket, making sure that it is not compromised. The cloud bucket owner should carefully configure her cloud bucket to avoid the issue we found and other misconfiguration flaws. In this case, an automatic configuration checker could be helpful. Most importantly, the cloud provider does have the responsibility to move more aggressively on detecting and removal of Bars from their systems. This, however, is non-trivial, given the privacy concern and the fact that some Bars can only be considered to be malicious by looking at the malicious activities they are involved in, such as those hosting Phishing pictures. Further research is needed to better

understand what the provider can do to address the issue.

Ethical issues. Most findings of the paper were made through analyzing the data crawled from the public domain. Regarding the study on the misconfiguration problem we found, our scanner was designed to minimize the privacy impacts on vulnerable repositories: specifically, it only tried out the functionality like file listing, uploading and downloading. The impact of such operations are very much in line with those of running online web testing tools (e.g., Sucuri [112]) on others' websites. Most importantly, we did this with the full intention to protect such repositories from future exploits, and also carefully avoided changing any existing content there and deleted from our system all the files downloaded. Further, we have already contacted the major vendors such as Groupon and the cloud providers like Amazon about those security breaches, and will continue to notify others and help them fix the configuration problem. So far, Groupon has acknowledged the importance of our findings and expressed gratitude for our help.

5.5 Summary

The emergence of using cloud repositories as a malicious service presents a new challenge to web security. This new threat, however, has not been extensively studied and little is known about its scope and magnitude and the techniques the adversary employs. In this chapter, we report the first systematic study on malicious and compromised cloud repositories and the illicit online activities built around them. We collected a small set of seeding Bars and identified a set of collective features from the websites connecting to them. These features describe the effort made by the adversary to protect Bars and utilize them to quickly build up a large campaign. Using these features, we developed a new scanner that detected over 600 Bars on top-of-the-line cloud platforms, including Google, Amazon, and others. Over these Bars, we performed a large-scale measurement study that led to surprising findings

of such attacks. Examples include the central roles those buckets play at each stage of a web attack (redirection, displaying Phishing content, exploits, attack payload delivery, etc.), the strategy to separate malware code and configuration files to avoid detection, and a configuration flaw never reported before that was likely exploited to compromise many cloud buckets. Our findings made an important step toward better understanding and effective mitigating of this new security threat.

CHAPTER VI

EFFICIENT PROMOTIONAL-INFECTION DETECTION THROUGH SEMANTIC INCONSISTENCY SEARCH

6.1 Introduction

Imagine that you google the following search term: `site:stanford.edu pharmacy`. Figure 27 shows what we got on October 9, 2015. Under the domain of Stanford University are advertisements (ad) for selling cheap viagra! Using various search terms, we also found the ads for prescription-free viagra and other drugs under `nidcr.nih.gov` (National Institute of Dental and Craniofacial Research), counterfeit luxury handbag under `dap.dau.mil` (Defense Acquisition Portal), and replica Rolex under `nv.gov`, the domain of the Nevada state government. Clearly, all those FQDNs have been unauthorizedly changed for promoting counterfeit or illicit products. This type of attacks (exploiting a legitimate domain for underground advertising) is called *promotional infection* in our research. Promotional infection is an attack exploiting the weakness of a website to promote content. It has been used to serve various malicious online activities (e.g., black-hat search engine optimization (SEO), site defacement, fake antivirus (AV) promotion, Phishing) through various exploit channels (e.g., SQL injection, URL redirection attack and blog/forum Spam). Unlike the attacks hiding malicious payloads (e.g., malware) from the search engine crawler, such as a drive-by download campaign, the promotional attacks never shy away from search engines. Instead, their purpose sometimes is to leverage the compromised domain's reputation to boost the rank of the promoted content (either what is directly displayed under the domain or the doorway page pointed by the domain) in the search results returned to the user when content-related terms are included in

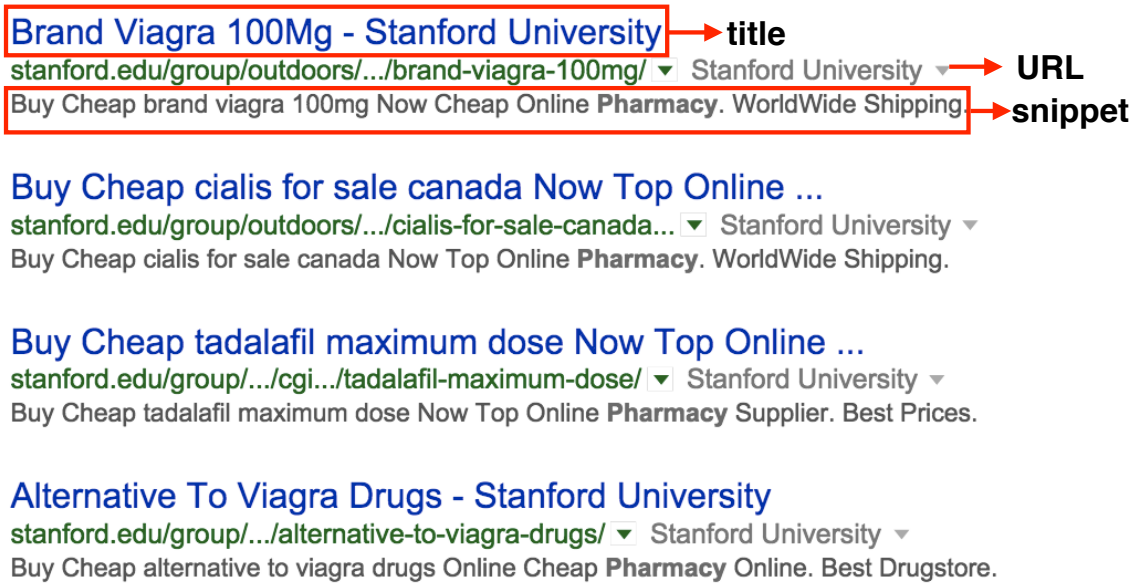


Figure 27: Search findings of promotional injections in stanford.edu. Search engine result is organized as title, URL and snippet.

her query. Such infections can inflict significant harm on the compromised websites through loss in reputation, search engine penalty, traffic hijacking and may even have legal ramifications. They are also pervasive: as an example, a study shows that over 80% doorway pages involved in black-hat SEO are from injected domains [109].

Catching promotional infections: challenges. Even with the prevalence of the promotional infections, they are surprisingly elusive and difficult to catch. Those attacks often do not cause automatic download of malware and therefore may not be detected by virus scanners like VirusTotal and Microsoft Forefront. Even the content injected into a compromised website can appear perfectly normal, no difference from the legitimate ads promoting similar products (e.g., drugs, red wine, etc.), ideological and religious messages (e.g., cult theory promotion) and others, unless its semantics has been carefully examined under the context of the compromised site (e.g., selling red wine is unusual on a government's website). So far, detection of the promotional infections mostly relies on the community effort, based upon the discoveries made by human visitors (e.g., PhishTank [12]) or the integrity checks that a compromised

website's owner performs. Although attempts have been made to detect such attacks automatically, e.g., through a long term monitoring of changes in a website's DOM structure to identify anomalies [34] or through computer vision techniques to recognize a web page's visual change [35], existing approaches are often inefficient (requiring long term monitoring or analyzing the website's visual effects) and less effective, due to the complexity of the infections, which, for example, can introduce a redirection URL indistinguishable from a legitimate link or make injected content only visible to the search engine.

Semantic inconsistency search. As mentioned earlier, fundamentally, promotional infections can only be captured by analyzing the semantic meaning of web content and the context in which they appear. To meet the demand for a large-scale online scan, such a semantic analysis should also be fully automated and highly efficient. Techniques of this type, however, have never been studied before, possibly due to the concern that a semantic-based approach tends to be complicated and less accurate. In this paper, we report a design that makes a big step forward on this direction, demonstrating it completely possible to incorporate Natural Language Processing (NLP) techniques into a lightweight security analysis for efficient and accurate detection of promotional infections. A key observation here is that for the attacks in Figure 27, inappropriate content shows up in the domains with specific meanings: no one expects that a `.gov` or `.edu` site promotes prohibited drugs, counterfeit luxury handbags, replica watches, etc. Such inconsistency can be immediately identified and located from the itemized *search result* on a returned search result page, which includes the title, URL and snippet for each result (as marked out in Figure 27). This approach, which detects a compromised domain (e.g., `stanford.edu`) based upon the inconsistency between the domain's semantics and the content of its result snippet reported by a search engine with regard to some search terms, is called *semantic*

inconsistency search or simply *SEISE*. Our current design of SEISE focuses on *sponsored top-level domain* (sTLD) like `.gov`, `.edu`, `.mil`, etc., that has a sponsor (e.g., US General Service Administration, EDUCAUSE, DoD Network Information Center), represents a narrow community and carries designated semantics (Section 6.2.1). Later we show that the technique has the potential to be extended to generic TLD (gTLD, see Section 6.4.2).

SEISE is designed to search for a set of strategically selected *irrelevant terms* under an sTLD (e.g., `.edu`) to find out the suspicious FQDNs (e.g., `stanford.edu`) associated with the terms, and then further search under the domains and inspect the snippets of the results before flagging them as compromised. To make this approach work, a few technical issues need to be addressed: (1) how to identify semantic inconsistency between injected pages and the main content of a domain; (2) how to control the false positives caused by the legitimate content including the terms, e.g., a health center sites on Stanford University (containing the irrelevant term "pharmacy"); (3) how to gather the search terms related to diverse promotional content. For the first issue, our approach starts with a small set of manually selected terms popular in illicit activities (e.g., gambling, drug and adult) and runs a *word embedding* based tool to calculate the semantic distance between these terms and a set of keywords extracted from the sTLD's search content, which describe the sTLD's semantics. Those most irrelevant are utilized for detection (Section 6.2.2). To suppress false positives, our approach leverages the observation that similar promotional content always appear on many different pages under a compromised domain for the purpose of improving the rank of the attack website pointed to by the content. As a result, a search of the irrelevant term under the domain will yield a result page on which many highly frequent terms (such as "no prescription", "low price" in the promotional content) turn out to rarely occur across the generic content under the same domain (e.g., `stanford.edu`). This is very different from the situation, for example, when a research article mentions

viagra, since the article will not be scattered across many pages under the site and tends to contain the terms also showing up in the generic content under the Stanford domain, such as "study", "finding", etc (Section 6.2.2). Finally, using the terms extracted from the result snippets of the sites detected, SEISE further automatically expands the list of the search terms for finding other attacks (Section 6.2.3).

We implemented SEISE and evaluated its efficacy in our research (Section 6.3). Using 30 *seed* terms and 403 sTLDs (across 141 countries and 89 languages), our system automatically analyzed 100K FQDNs and along the way, expanded the keyword list to 597 terms. In the end, it reported 11K infected FQDNs, which have been confirmed to be compromised[1] through random sampling and manual validation. With its low false detection rate (1.5%), SEISE also achieved over 90% detection rate. Moving beyond sTLD, we further explore the potential extension of the technique to gTLDs such as `.com` (Section 6.4.2). A preliminary design analyzes `.com` domains using their site tag labeled by SimilarSites [15], which is found to be pretty effective: achieving a false detection rate (FDR) of 9% when long keywords gathered from compromised sTLDs are used.

Our findings. Looking into the promotional infections detected by SEISE, we were surprised by what we found: for example, about 3% (175) of `.gov` domains and 3% (246) of `.edu` domains are injected; also around 2% of the 62,667 Chinese government domains (`.gov.cn`) are contaminated with ads, defacement content, Phishing, etc. Of particular interest is a huge gambling campaign we discovered (Section 6.4.3), which covers about 800 sTLDs and 3000 gTLDs across 12 countries and regions (US, China, Taiwan, Hong Kong, Singapore and others). Among the victims are 20 US academia institutes such as `nyu.edu`, `ucsd.edu`, 5 government agencies like `va.gov`,

[1]Note that in line with the prior research [71], the term "compromise" here refers to not only direct intrusion of a web domain, which was found to be the most common cases in our research (80%, see Section 6.5), but also posting of illicit advertising content onto the domain through exploiting its weak (or lack of) input sanitization: e.g., blog/forum Spam and link Spam (using exposed server-side scripts to dynamically generate promotion pages under the legitimate domain).

`makinghomeaffordable.gov`, together with 188 Chinese universities and 510 Chinese government agencies. We even recovered the attack toolkit used in the campaign, which supports automatic site vulnerability scan, shell acquisition, SEO page generation, etc. Also under California government's domain `ca.gov`, over one thousand promotion pages were found, all pointing to the same online casino site. Another campaign involves 102 US universities (`mit.edu`, `princeton.edu`, `stanford.edu`, etc.), advertising "buy cheap essay". The scope of these attacks go beyond commercial advertising: we found that 12 Chinese government and university sites were vandalized with the content for promoting Falun Gong. Given the large number of compromised sites discovered, we first reported the most high-impact findings to related parties (particularly universities and government agencies) and will continue to do so (Section 6.5).

Further, our measurement study shows that some sTLDs such as `.edu`, `.edu.cn` and `.gov.cn` are less protected than the `.com` domains with similar Alexa ranks, and therefore become soft targets for promotional infections (Section 6.4.2). By effectively detecting the attacks on these sTLDs, SEISE raises the bar for the adversary, who has to resort to less guarded gTLDs, which typically have much lower Alexa ranks, making the attacks, SEO in particular, less effective.

Contributions. The contributions of the paper are outlined as follows:

• *Efficient semantics-based detection of promotional infections.* We developed a novel technique that exploits the semantic gap between domains (sTLDs in particular) and unauthorized content they host to detect the compromised websites that serve underground advertising. Our technique is highly effective, incurring low false positives and negatives. Also importantly, it is simple and efficient: often a compromised domain can be detected by querying Google no more than 3 times. This indicates that the technique can be easily scaled, with the help of search providers.

• *Measurement study and new findings.* We performed a large-scale measurement

study on promotional infections, the first of this kind. Our research brings to light several high-impact, ongoing underground promotion campaigns, affecting leading educational institutions and government agencies, and the unique techniques the perpetrator employs. Further we demonstrate the impacts of our innovation, which significantly raises the bar to promotional infections and can potentially be extended to protect generic domains.

6.2 SEISE: Design

As mentioned earlier, promotional infections often do not propagate malicious payloads (e.g., malware) directly and instead only post ads or other content that legitimate websites may also contain. This makes detection of such attacks extremely difficult. In our research, we look at the problem from a unique perspective, the inconsistency between the malicious advertising content and the semantics of the website, particularly, what is associated with different sTLDs. More specifically, underlying SEISE are a suite of techniques that search sTLDs (`.edu`, `.gov`, etc.) using *irrelevant bad terms* (*IBT*) (the search terms unrelated to the sTLDs but heavily involved in malicious activities like Spam, Phishing) to find potentially infected FQDNs, analyze the context of the IBTs under those FQDNs to remove false positives and leverage detected infections to identify new search terms, automatically expanding the IBT list. Below we elaborate on this design.

6.2.1 Overview

Architecture. Figure 28 illustrates the architecture of SEISE, which includes *Semantics Finder*, *Inconsistency Searcher*, *Context Analyzer* and *IBT Collector*. Semantics Finder takes as its input a set of sTLDs, automatically identifying the keywords that represent their semantics. These keywords are compared with a seed set of IBTs to find the most irrelevant terms. Such selected terms are then utilized by Inconsistency Searcher to search related sTLDs for the FQDNs carrying these terms.

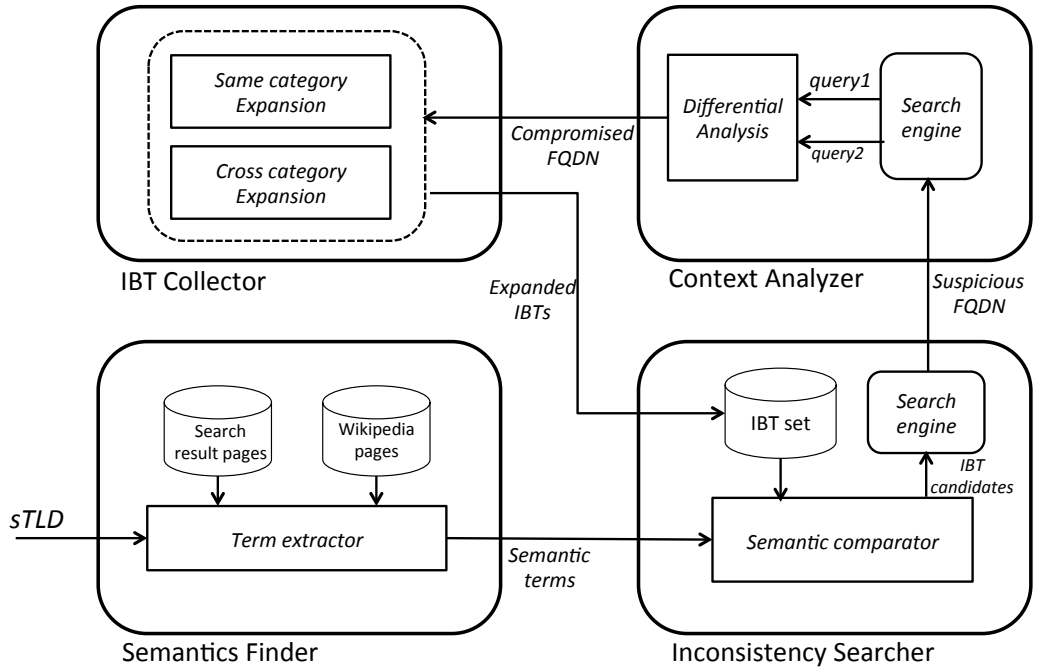


Figure 28: Overview of the SEISE infrastructure.

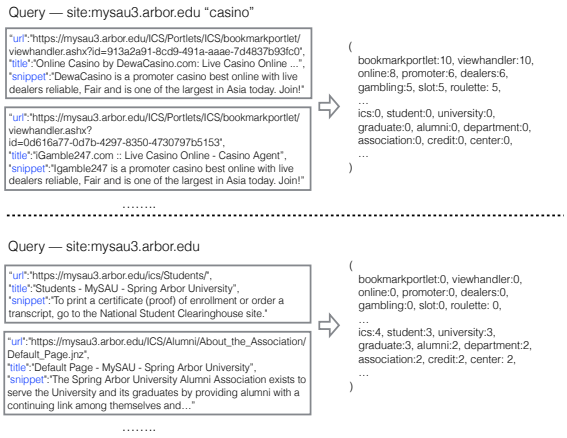
Under each detected FQDN, Context Analyzer further evaluates the context of discovered IBTs through a differential analysis to determine whether after removing *stop words*, i.e., the most common words like 'the' from the context, frequently-used terms identified there (e.g., the search result of `site:stanford.edu pharmacy`) become rare across the generic content of the FQDN (e.g., the search result of `site:stanford.edu`), which indicates that the FQDN has indeed been compromised. Such FQDNs are reported by SEISE and their snippets are used by IBT Collector to extract keywords. Those with the largest semantic distance from the sTLDs are added to the IBT list for detecting other infected FQDNs.

Example. To explain how SEISE works, let us take a look at the example at the beginning of the paper (Figure 27). For the sTLD `.edu`, SEISE first runs *Semantics Finder* to automatically extract keywords to profile sTLD, e.g., "education", "United States" and "student". In the meantime, a seed set of IBTs, including "casino", "pharmacy" and others, are converted into vectors using the word-embedding technique. Their semantic gap with the `.edu` sTLD is measured by calculating the *cosine*

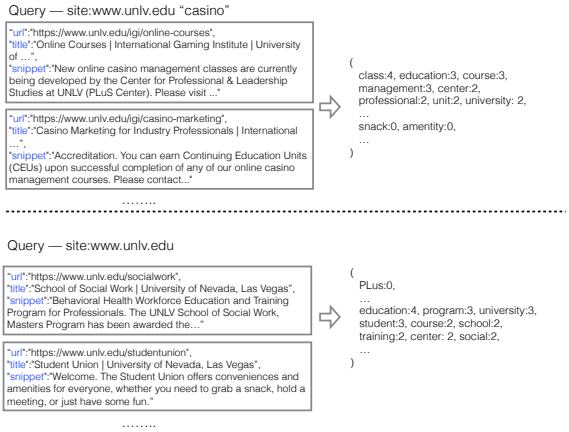
distances between individual terms (like "pharmacy") and the sTLD keywords (such as "education", "United States" and "student"). It turns out that the terms like "`pharmacy`" are among the most irrelevant (i.e., with a large distance with `.edu`). It is then used to search Google under `.edu`, which shows the FQDN `stanford.edu` hosting the content with the search term. Under this FQDN, SEISE again searches for "`pharmacy`." The results page is presented in Figure 27. As we can see, many search result items (for different URLs) contain same topic words, similar snippet and even URL patterns, which are typically caused by mass injection of unauthorized advertising materials. These items form the *context* for the IBT "`pharmacy`" in `stanford.edu`.

Our approach then converts the context (the result items) found into a high-dimensional vector, with the frequency of each word (except those common *stop words* like 'she', 'does', etc.) as an element of the vector. The vector, considered to be a representative of the context, then goes through a differential analysis: it is compared with the vector of a *reference*, the search results page of `site:stanford.edu` that describes the generic content under the FQDN. The purpose is to find out whether the context is compatible with the theme of the FQDN. If the distance between them is large, then we know that this FQDN hosts a large amount of similar text semantically incompatible with its theme (i.e., most of the high frequent words in the suspicious text, such as "viagra", rarely appear in the common content of the FQDN). Also given the fact that such text is the context for the search terms irrelevant to the sTLD of the current FQDN but popular in promotional infections, we conclude that the FQDN `stanford.edu` is indeed compromised.

Once an infection is detected, the terms extracted from the context of "`pharmacy`" are then analyzed and those most irrelevant to the semantics of `.edu` are added to the IBT list for finding other compromised FQDNs. Examples of the terms include "viagra", "cialis", and "tadalafil". In addition to the words, the URL pattern of the



(a) **Differential analysis of an injected site.** Cosine distance = 0.97



(b) **Differential analysis of a non-injected site.** Cosine distance = 0.14

Figure 29: Differential analysis of an injected site and a non-injected site.

infection is then generalized to detect other advertising targets (e.g., red wine) not included in the initial IBT list (e.g., those for promoting illegal drugs). The same technique can also be applied to find out compromised gTLDs like the `.com` FQDNs involved in the same campaign.

6.2.2 Semantics-based Detection

In this section, we present the technical details for *Semantics Finder*, *Inconsistency Searcher* and *Context Analyzer*.

Finding semantics for sTLDs. The first step of our approach is to automatically build a semantic profile for an sTLD. Such a profile is represented as a set of terms,

which serve as an input to the Inconsistency Searcher for choosing right IBTs. For example, the semantic representation of the sTLD `.edu.cn` could be "Chinese university", "education", "business school", etc. SEISE automatically identifies these terms from different sources using a *term extraction* technique. Specifically, the following two sources are currently utilized by our prototype:

• *Wikipedia*: the Wikipedia pages for sTLDs provide a comprehensive summary of different sTLDs. For example, `https://en.wikipedia.org/wiki/.mil` profiles the sTLD `.mil`, including its sponsor ("DoD Information System Agency"), intended use ("military entities"), registration restrictions ("tightly restricted to eligible agencies"), etc. In our research, we ran a crawler that collected the wiki pages for 80 sTLDs.

• *Search results*: the search results page for an sTLD query (e.g., `site:gov`) lists high-profile websites under the sTLD. As mentioned earlier, each search result includes a snippet of a website, which offers a concise but high-quality description of the website. Since the websites under the sTLD carry the semantic information of the sTLD, such descriptions can be used as another semantic source of the sTLD. Therefore, our approach collected the search result pages of all 403 sTLDs using automatically-generated queries in the form of "site:sTLD", such as `site:edu`. From each result page, top 100 search results are picked up for constructing the related sTLD's semantic profile.

From such sTLD semantics sources, the Semantics Finder runs a content term extraction tool to automatically gather keywords from the sources. These keywords are supposed to best summarize the topic of each source and therefore represent the semantics of an sTLD. In our implementation, we utilized an open-source tool *topia.termextract* [19] for this purpose. From each keyword extracted, our approach further calculates its frequency, which is assigned to the keyword as its *weight*. All together, top 20 keywords are chosen for each sTLD as its semantics profile.

A problem is that among all 403 sTLDs, 71 of them are non-English ones, which

include Chinese, Russian, French, Arabic, etc., 89 languages altogether. Analyzing these sTLDs in their native languages is complicated, due to the challenges in processing these languages: for example, segmenting Chinese characters into words is known to be hard [123]. To solve this problem, we utilized Google Translate to convert the search page of an non-English sTLD query into English and then extract their English keywords. The approach was found to work effectively, capturing non-English promotional infections (see Section 6.4).

Searching for inconsistency. The Inconsistency Searcher is designed to find out the IBTs with great semantic gaps with a given sTLD, and use the terms to search the sTLD for suspicious (potentially compromised) FQDNs. To this end, we first selected a small set of seed IBTs as an input to the system. These IBTs were collected from spam trigger word lists [23, 24] and SEO competitive word list [25], which are popular terms used in counterfeit medicine selling, online gambling and Phishing. From those terms, the most irrelevant ones are picked up for analyzing a given sTLD. Such terms are found by comparing them with the semantics profile of the FQDN, that is, the set of keywords output by the Semantics Finder.

Specifically, such a semantic comparison is performed by SEISE using a word-embedding tool called `word2vec` [21], a neural network that builds a vector representation for each term by learning from the context in which the term occurs. In our research, we utilized the English Wikipedia pages as the context for each term to compute its vector and measure the distance between two words using their vectors. In this way, the IBTs irrelevant to a given sTLD can be found and used to search under the FQDN for detecting the suspicious ones. The approach works as follows:

• We downloaded all 30 GB Wikipedia pages and ran a program to preprocess those pages by removing tables and images while preserving their captions. Individual sentences on the pages were further tokenized into terms using a phrase parser.

• Given an input term (an IBT or a keyword in the sTLD's semantics profile), our

approach runs `word2vec` to train a skip-gram model, which maps the term into a high-dimensional vector $\langle d_1, d_2, ...d_i, ...\rangle$ to describes the term's semantics. This vector is generated from all the sentences involving the term, with individual elements describing the term's relations with other terms in the same sentence across all such sentences in the Wikipedia dataset.

• Given the vectors of an IBT and an sTLD keyword, our approach measures the semantic distance between them by calculating the cosine distance between their vectors. For each IBT, its average distance to all the keywords is used to determine its effectiveness in detecting promotional infections. In our research, we found that when the distance becomes 0.6 (at least 20 terms are still there within our seed set) or more, almost no compromised site is missing (see Figure 31(a) in Section 6.4). The IBTs selected according to such a threshold are then sent to the search engine together with the sTLD through the query `site:sTLD+IBT` (e.g.,`site:edu casino`). From the search result page, top 100 items (URLs) are further inspected by the Context Analyzer to determine whether related FQDNs are indeed compromised, which is detailed in the followed subsection.

As an example, again, let us look at Figure 29: in this case, the IBT "`casino`" has a distance of 0.72 with regard to the semantics of `.edu` and therefore was run under the sTLD; from the search pages, top FQDNs, including `mysau3.arbor.edu`, `www.unlv.edu`, were examined to detect compromised FQDNS.

Analyzing IBT context. As mentioned earlier, even the terms most irrelevant to an sTLD could show up on some of its pages for a legitimate reason. For example, the word 'casino' has a significant semantic distance with the sTLD `.edu`, which does not mean, however, that the `.edu` sites cannot carry a poster about one's travel to Las Vegas or a research article about a study on the gambling industry. Actually, a direct search of the term `site:edu casino` yields a result page with some of the items being legitimate. To identify those compromised FQDNs, the Context Analyzer

automatically examines the individual FQDN on the result page, using a *differential analysis* (Figure 28) to detect those truly compromised.

More specifically, the differential analysis involves two independent queries, one on the suspicious FQDN together with the IBT (e.g., `site:life.sunysb.edu casino`) and the other on the FQDN alone (e.g., `site:life.sunysb.edu`) whose results page serves as the reference. The idea is based on the observation that in a promotional infection, the adversary has to post similar text on many different pages (sometimes pointing to the same site) for promoting similar products or content. This is necessary because the target site's rank needs multiple highly-ranked pages on the compromised site to promote. The problem for such an attack is that the irrelevant content, which is supposed to rarely appear under the FQDN, becomes anomalously homogenous and pervasive under a specific IBT. As a result, when we look at the search results of the IBT under the FQDN, their URLs and snippets tend to carry the words rarely showing up across the generic content (i.e., the reference) with much higher frequencies than their accidental occurrences under the FQDN. On the other hand, in the case of legitimate content including the IBT, the search results (for the IBT under the FQDN) will be much more diverse and the words involved in the IBT's context often appear on the reference and are compatible with the generic content of the site; even for the irrelevant terms in the context, their frequencies tend to be much lower than those in the malicious context. This is because it is unlikely that the term irrelevant to the theme of the site accidentally appears in similar context across many pages, which introduces an additional set of highly-frequent irrelevant terms. As an example, let us look at Figure 29(a) that shows a compromised FQDN and Figure 29(b) that illustrates a legitimate FQDN. The highly-frequent words extracted from the former under the IBT 'casino', such as 'bookmarkporlet', 'dealers', 'slot', never show up across the URLs and snippets of the reference that represents the generic content of the FQDN (the result of the query `site:mysau3.arbor.edu`). In contrast, a query

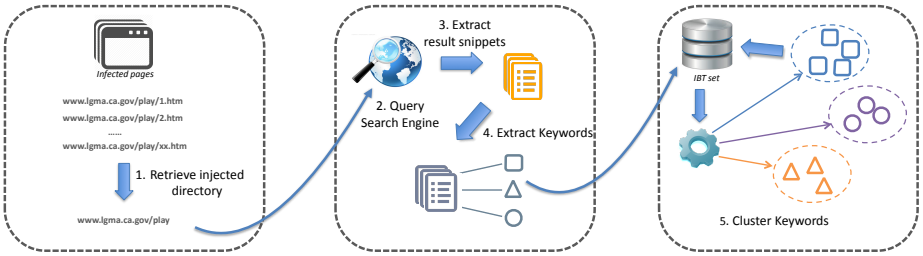


Figure 30: IBT SET Extension. The process to find IBTs in new category consists of five steps: Injected URLs are collected to find the injected directory path (❶). Then, the injected directory path is used as search keyword, i.e., site:www.lgma.ca.govplay to list more search result items (❷). After fetching search result snippets(❸), critical terms are extracted (❹), and those that show semantics irrelevance are filtered for clustering (❺). Once a new cluster is formed, we manually check and label it with its semantics.

of the legitimate FQDN using the same IBT yields a list of results whose URLs and snippets have highly diverse content, with some of their words also included in the generic content, such as 'class', 'education' and 'university', and most others (except the IBT itself) occurring infrequently.

To compare the two search result pages for identifying the truly compromised site, the Context Analyzer picks up top 10 search results from each query and converts them into a high dimensional vector. Specifically, our approach focuses on the URL and the content snippet for each result item. We segment them into words using delimiters such as space, comma, dash, etc., and remove *stop words* (those extremely common words like 'she', 'do', etc.) using a stop word list [16]. In this way, each search item is tokenized and the frequency of each token, across all 10 results is calculated to form a vector $V =< w_0, w_1, ...w_i, ... >$, where w_i is the frequency of a word corresponding to that position. For the two vectors V_b (the search page under the IBT) and V_g (the reference, that is, the search page of the FQND without the IBT), SEISE calculates their Cosine distance: $1 - \frac{V_b \cdot V_g}{\|V_b\| \|V_g\|}$.

In Figure 29(a), the distance of the vector for the IBT 'casino' with the reference vector is 0.97. In Figure 29(b), where the FQDN is not compromised, we see that the vector under the IBT 'casino' is much closer to that of the reference, with a distance

of 0.14. In our research, we chose 0.9 as a threshold to parameterize our system: whenever the Cosine distance between the results of querying an FQDN under an IBT and the reference of the FQDN goes above the threshold, the Context Analyzer flags it as infected. This approach turns out to be very effective, incurring almost no false positives, as elaborated in Section 6.3.

Discussion. SEISE is carefully designed to work on search result pages instead of the full content of individual FQDNs. This is important because the design helps achieve not only high performance but also high accuracy. Specifically, a semantic analysis on a small amount of context information (title, URL and snippet of a search result) is certainly much more lightweight than that on the content of each web page. Also interestingly, focusing on such context helps avoid the noise introduced by the generic page content, since the snippet of each search result is exactly the text surrounding an IBT, the part of the web page most useful for analyzing the suspicious content it contains. In other words, our approach leverages the search engine to zoom in on the context of the IBT, ignoring unrelated content on the same web page.

6.2.3 IBT SET Extension

A critical issue for the semantic-based detection is how to obtain high-quality IBTs. Those terms need to be malicious and irrelevant to the semantics of an sTLD. Also importantly, they should be diverse, covering not only different keywords the adversary may use in a specific category of promotional infections, like unlicensed pharmacy, but also those associated with the promotional activities in different categories, such as gambling, fake product advertising, academic cheating, etc. Such diversity is essential for the detection coverage SEISE is capable of achieving, since a specific type of promotional attack (e.g., fake medicine) cannot be captured by a wrong IBT (e.g., 'gambling').

As mentioned earlier, the seed IBT set used in our research includes 30 terms,

which were collected from several sources, including spam trigger word lists [23, 24] and SEO competitive word list [25]. These IBTs are associated with the attacks such as blackhat SEO, fake AV and Phishing. To increase the diversity of the set, SEISE expands it in a largely automated way, both within one category and across different categories. More specifically, our approach leverages NLP techniques to gather new IBTs from the search items reported to contain malicious content, and further cluster these IBTs to discover new categories. Here we elaborate on this design.

Finding IBTs within a category. Once a compromised FQDN has been identified using an IBT, the search results that lead to the detection (for the query "`site: FQDN+IBT`") can then be used to find more terms within the IBT's category. This is because the result items are the context of the IBT, and therefore include other bad terms related to the IBT. Specifically, similar to the Semantics Finder, the IBT Collector runs the term extraction tool on each result item, including its title, URL and snippet, to gather the terms deemed important to the context of the IBT. Such terms are further inspected, automatically, against the semantics of an sTLD by measuring their average distances with the keywords of the FQDN (that is, converting each of them into a vector using `word2vec` and then calculating the Cosine distance between two vectors). Those sufficiently away from the FQDN's semantics (with a distance above the aforementioned threshold) are selected as IBTs.

Finding new categories. Extracting keywords from the context of an IBT can only provide us with new terms in the same category. To detect the infections in other categories, we have to extend the IBT set to include the terms in other types of illicit promotions. The question is how to capture new keywords such as 'prescription-free antibiotic' that are distinguished from the IBTs in the known category such as 'gambling', 'casino', etc. A key observation we leveraged in our study is that the adversary sometimes compromises an FQDN to perform multiple types of advertising: depending on the search terms the user enters, an infected website may provide different

kinds of promotional content, for drug, alcohol, gambling and others. Further the ads serving such a purpose are often deposited under the same directory, along the same path under a compromised FQDN. This enables us to exploit the URL included in a contaminated result item (as detected by SEISE) to find the promotional materials unrelated to the context of the IBT in use.

Specifically, from each flagged FQDN, the IBT Collector first picks up all the URLs leading to malicious content, and from them, identifies the most commonly shared path under the FQDN. For example, from the URLs `www.lgma.ca.gov/play/popular/1*.html`, `www.lgma.ca.gov/play/home/2*.html` and `www.lgma.ca.gov/play/club/3*.html` (detected using the IBT 'casino'), the shared path under the FQDN is `www.lgma.ca.gov/play`. Using this path, our approach queries Google again with '`site:FQDN+path`': e.g., `site:www.lgma.ca.gov/play`. From the results page of the query, critical terms are extracted by analyzing snippets under individual result items. These terms are further compared with the semantics of the current sTLD: those most irrelevant (with a cosine distance above the threshold 0.9) are kept. Finally, the vectors of these terms are clustered using the classic k-Nearest-Neighbor (k-NN) algorithm (with $k = 10$) together with all existing IBTs. Once a new cluster is formed in this way, we manually look at the cluster and label it with its semantics (gambling, drug selling, academic cheating, etc.). Note that this manual step is just for labeling, not for adjusting the clustering outcomes, which were found to be very accurate in our research (Section 6.3.3).

In the above example as illustrated in Figure 30, the query `site:www.lgma.ca.gov/play` leads to the search results page. From the items on the page, the IBT Collector automatically recovers a set of critical terms, including 'goldslot', 'payday loan', 'cheap essay' and others. Clustering these terms, some of them are classified into existing categories such as gambling, drug, etc., while the rest are grouped into a new cluster, containing 'cheap essay', 'free term paper' along with other 15 terms.

This new cluster is found to be indeed a new attack category, and labeled as 'academia cheating'. In our research, we ran the approach to extend our IBT set, from 30 terms to 597 effective terms, from 3 categories (gambling, drug, etc.) to 10 large categories (financial, cheating, politics, etc.). Our manual validation shows that the results are mostly correct.

6.3 Implementation and Evaluation

In this section, we report our implementation of SEISE and evaluation of its efficacy. Our study show that the simple semantics-based approach works well in practice: it automatically discovered IBTs, achieved an low false detection rate (1.5%) at over 90% of coverage and also captured 75% infected domains never reported before (Section 6.3.3).

6.3.1 Implementing SEISE

The design of SEISE (Section 6.2) was implemented into a prototype system, on top of a set of building blocks. Here we briefly describe these nuts and bolts and then show how they are assembled into the system.

Nuts and bolts. Our prototype system was built upon three key functional components, *term extractor*, *static crawler* and *semantic comparator*. Those components are extensively reused across the whole system, as illustrated in Figure 28. They were implemented as follows:

• Term extractor accepts text as its input, from which it automatically identifies critical terms. The component was implemented in Python using an open-source tool *topia.termextract*.

• Static crawler accepts query terms, looks for the terms through search engines and returns results with a pre-determined number of items. In our implementation, the crawler was developed in Python and utilized the Google Web Search API [10] and the Bing Search API [3] to get search results.

• Semantic comparator accepts a set of terms and compares them with the keywords of an input sTLD. It can return the average distance of each term with those keywords or the terms whose distances are above a given threshold. This component was implemented as a Python program that integrates the open-source tool `word2vec`. As mentioned earlier, we trained the language model used by `word2vec` with the whole Wikipedia dataset, from which our implementation automatically collected the context for each term before converting it to a high-dimensional vector.

System building. Using these building blocks, we constructed the whole system as illustrated in Figure 28. Specifically, the Semantic Finder was developed to run the static crawler to gather the content under an sTLD and then call the term extractor to identify the keywords for the domain. The Inconsistency Searcher invokes the semantic comparator to determine the most irrelevant IBTs before using the crawler to search for the terms. The Context Analyzer includes a differential analyzer component implemented with around 300 lines of Python code. For each suspicious FQDN, the analyzer calls the crawler to query the search engine twice, one under an IBT and the other for getting the reference (the generic content). It reports the domain considered to be compromised. Finally, the IBT Collector uses the crawler to search for the selected URL path under the detected domain, then the extractor to get critical terms from the search results and the semantics comparator to find out new IBTs. Over these IBTs, we further integrated the k-NN module provided by the scikit-learn open source machine learning library [106] to cluster them and discover new bad-term categories.

6.3.2 Experiment Setting

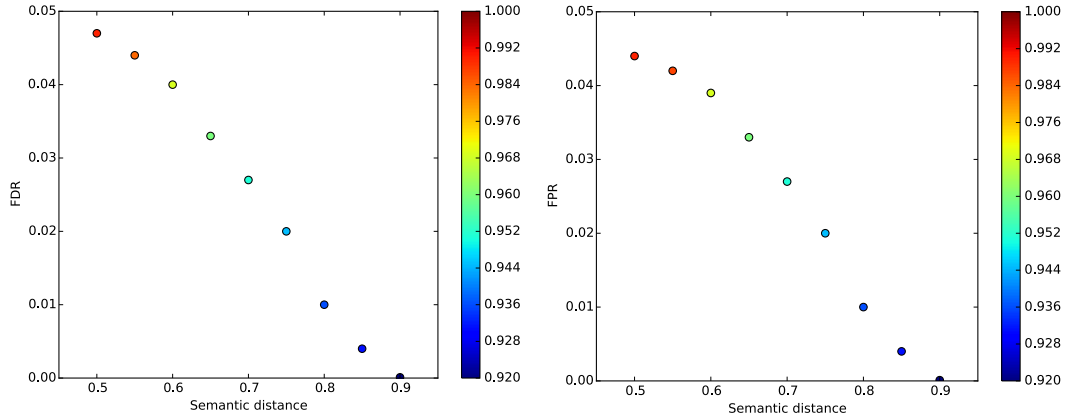
Data collection. To evaluate SEISE, we ran our prototype on three datasets: the labeled bad set and good set, and the unknown set including 100K FQDNs collected from search engines, using 597 search terms, as explicated below.

• *Bad set*. We collected the FQDNs confirmed to have promotional infections from *CleanMX* [42], a blacklist of compromised URLs. A problem here is that these URLs are associated with different kinds of malicious activities and it is less clear whether they are promotional infection. What we did is to collect all the sTLD URLs from the CleanMX feed from 2015/07 to 2015/08, and further manually inspected all these URLs. Specifically, whenever we saw that advertising, Phishing, defacement content showing up in the search results of a URL, it is considered to be exploited for promotional infections. We further classified these URLs into different categories and also manually identified related IBTs. In this way, we built a bad set with 300 FQDNs (together with 15 IBTs in three categories).

• *Good set*. Using the IBTs collected from the bad set, we further searched under the sTLDs for the FQDNs ("`site:sTLD+IBT`") that contained those terms but were not compromised. These domains were used to understand the false detections that could be introduced by SEISE. Altogether, we collected a good set of 300 FQDNs related to 15 IBTs and three categories.

• *Unknown set*. We gathered 403 sTLDs and manually selected 30 IBTs in three categories. Running these IBT seeds on these sTLDs, we crawled Google and Bing over three months, collecting 100K FQDNs. This dataset was used as the unknown set for discovering new promotional infections.

Resources and validation. In all our experiments, our prototype system was run within Amazon EC2 C4.8xlarge instances equipped with Intel Xeon E5-2666 36 vCPU and 60GiB of memory. To collect the data for the unknown set, we deployed 20 crawlers within virtual machines with different IP settings. These crawlers utilized the APIs provided by Google and Bing to dump the outcomes of the queries, from 2015/08 to 2015/10.



(a) **False detection rate in different semantics distances.** Color bar shows the coverage rate.

(b) **False positive rate in different semantics distances.** Color bar shows the coverage rate.

Figure 31: Evaluation results on good set and bad set.

To validate the findings made on the unknown set, we employed a methodology that combined anti-virus (AV) scanning, blacklist checking and manual analysis. Specifically, for the FQDN reported by our system, we first scanned their URLs with VirusTotal and considered that the URLs were indeed suspicious when at least two scanners flagged the domain. Then, all such suspicious URLs were cross-checked against the blacklist of CleanMX. For those confirmed by both VirusTotal and CleanMX, their FQDNs were automatically labeled as compromised. For other domains also detected by SEISE, we randomly sampled 20% of them and manually checked whether they were indeed compromised.

6.3.3 Evaluation Results

Over the aforementioned datasets, we thoroughly evaluated our prototype. Our study shows that SEISE is highly effective: it achieved near zero *False Detection Rate* (FDR, i.e., FP/(FP+TP)) and over 90% coverage (i.e., TP/(TP+FN)) or below 4.7% FDR, 4.4% *False Positive Rate* (FPR, i.e., FP/(FP+TN)) and nearly 100% coverage on the labeled sets (the bad and good set); with the threshold chosen to balance FDR and FPR, we further ran SEISE over the unknown set, which reported over 11K

compromised sites, with an FDR of 1.5% and a coverage over 90%. Also importantly, 75% of infections discovered from the unknown set are likely *never reported before*, including 3 large-scale campaigns, on which we elaborate in Section 6.4. All these findings were made in a highly efficient and scalable way: on average, only 2.3 queries were made for finding a new compromised FQDN and the delay caused by analyzing the query results and other computing resources consumed for this purpose were completely negligible.

Accuracy and coverage. We evaluated the accuracy and the coverage of SEISE under a given set of IBTs. In this case, what can be achieved are all dependent on the Context Analyzer, which ultimately decides whether to flag an FQDN as compromised. In our research, we first studied our system over the labeled good set and bad set, and then put it to test over the unknown set. Figure 31(a) and 31(b) illustrate the results over the labeled sets, in response to different thresholds for semantic distances (between the reference and the query of an IBT). As we can see here, when the threshold goes up, the FDR goes down and so does the coverage. On the other hand, loosening the threshold, which means that the IBT is becoming less irrelevant to the semantics of the sTLD, improves the coverage, at the cost of the FDR. Overall, the results show that SEISE is highly accurate: by setting the threshold to 0.9, we observe almost no false detection (FDR: 0.5% and FPR: 0.4%) with a 92% of coverage; alternatively, if we can tolerate 4.7% FDR (FPR: 4.4%), the coverage becomes close to 100%. In our research, the threshold 0.9 was then utilized to analyze the unknown set.

On the unknown set, we ran SEISE to query 597 IBTs under 403 sTLDs. Our prototype inspected 100K FQDNs in total. 11,473 of them were flagged as compromised, about 11% of the whole unknown set. Table 21 and Table 22 summarize our findings, which are further discussed in Section 6.4. Among all that were detected, 3% were confirmed by both VirusTotal [20] and CleanMX [42], 22% were found by at least one

Table 20: Number of IBTs in each round.

Round	# of categories	# of IBTs per category	Avg. length
0	3	10	2.6
5	5	18	3.0
10	8	25	3.1
15	10	40	3.2
20	10	60	3.8

of these two AV systems and further validated manually, and 1000 of the remaining were inspected manually. All together, the FDR measured from the unknown set is as low as 1.5%. We further randomly sampled 500 result pages related to 10 categories of IBTs and found that our prototype reported 53 infections and missed 5, which indicates a coverage of about 90%. Also, note that over 75% of the infections have never been reported (missed by both VirusTotal and CleanMX). We have reported the most prominent ones among them to related organizations and are helping them fix the problem, and will continue to work on other cases.

IBT expansion. The effectiveness of SEISE also relies on its capability to discover new IBTs and find new attack instances across different categories. As discussed before, our prototype starts with a small set of seed IBTs, 30 terms in three categories. After searching for all these terms under all the sTLDs, a set of compromised FQDNs are detected, which are further used by the IBT Collector to extract new terms for searching all 403 sTLDs again. In our research, we repeated such iteration 20 times, expanding the IBT set to 597 terms and 10 categories. All the terms and categories were manually confirmed to be correct. Table 20 presents the numbers for the terms and the categories, together with examples of new terms detected, after the 1st, 5th, 10th, 15th and 20th iterations. As we can see here, the number of categories and number of IBTs increase quickly (with a increase rate of 60% and 180%, respectively) in the first 10 iterations, which indicate that our IBT expansion method is efficient for both in-category and cross-category expansion. Also, Table 22 illustrates the total categories of IBTs flagged by SEISE after these iterations.

Table 21: Top 10 sTLDs with most injected domains.

sTLD	Est. total	# monitored	# injected	Volume	Injected size
gov.cn	62,667	2,904	1,240	12%	FQDN: 1,840 URL: 172,244
edu.vn	16,148	2,032	262	3%	FQDN: 312 URL: 22,543
edu	8,955	2,502	246	3%	FQDN: 250 URL: 29,580
edu.cn	3,912	1,173	238	2%	FQDN: 403 URL: 34,308
edu.au	9,594	1,968	204	2%	FQDN: 223 URL: 21,563
gov.co	-	1,892	200	2%	FQDN: 253 URL: 23,022
gov	6,251	1,562	175	2%	FQDN: 178 URL: 15,720
gov.in	4,272	1,402	141	1%	FQDN: 163 URL: 14,572
edu.in	3,892	1,243	132	1%	FQDN: 172 URL: 12,034
edu.mx	8,232	1,372	126	1%	FQDN: 144 URL: 11,056

Table 22: Categories of IBTs.

Category	Keyword			Injected site		
	# kw.	avg. len	example	# FQDN	# domains	Example
Gambling	62	3.5	casino, slot machine	3650	2134	ca.gov (Alexa: 649)
Drug	64	3.2	cheap xanax, no prescription	2047	1742	princeton.edu (Alexa: 3558)
General	83	3.4	nike air max, green coffee bean	1673	1572	nih.gov (Alexa: 196)
Cheating	52	4.2	fake driving permit, cheap essay	1107	1017	mit.edu (Alexa:789)
Financial	65	3.6	payday loan, quick loan	1092	947	nsf.gov (Alexa:16,303)
Travel	58	4.5	cheap airfare, hotel deal	972	924	gmu.edu (Alexa: 8058)
Luxury	59	3.2	cheap gucci, discounted channel	890	876	nv.gov (Alexa:25,875)
Adult	60	4.6	qvod, sex movie	922	843	tsinghua.edu.cn (Alexa: 6717)
Software	53	5.2	free download, system app	807	734	noaa.gov (Alexa:1126)
Politics	41	3.2	islamic state, falun gong	372	342	buaa.edu.cn (Alexa:33,807)

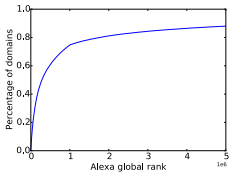
Performance. We further evaluated the performance of our prototype, in an attempt to understand the scalability of our design. We found that except the delay caused by receiving the results from Google, the overhead for analyzing search results and detecting compromised sites are exceedingly low: by running 10000 randomly selected queries (50 IBTs over 200 sTLDs), we observed that the average time for analyzing 1K result items, excluding the waiting time for the search engine, was 1ms, and also the memory and CPU usages stayed below 5% respectively. The main hurdle here is the delay caused by the search engine: for Google, it ranged from 5ms to 8ms per one thousand queries. The design of SEISE already limits the number of queries that needed to be made for detecting infected FQDNs: in the experiments, we found that on average, a compromised FQDN was detected after 2.3 term queries. We believe that by working with the search provider (Google, Bing etc.), SEISE can be easily scaled with a quick turnaround of the search results.

6.4 Measurement

Based upon what was detected by SEISE, we performed a measurement study to understand the promotional infections on sTLDs, particularly the semantic inconsistency these attacks introduce. Our study brings to light the pervasiveness of the attacks and their significant impacts, affecting the websites of leading academic institutions and government agencies around the world. Further discovered are a set of surprising findings and their insights, which have never been known before. For example, apparently sTLDs are soft targets for promotional infections, highly ranked and also easier to compromise compared with gTLD sites of similar ranks; as a result, by mitigating the threats to the sTLD domains, we raise the bar for the adversary, depriving him of easy access to the resources highly valuable to the promotional attacks, which rely on the compromised site's rank to boost the rating of malicious content. As another example, we show that semantic inconsistency can also be observed in the promotional infections on gTLDs such as `.com`, `.net`, etc., even though these domains tend to have a much more diverse semantic meaning. Based upon this observation, a preliminary exploration highlights the potential of extending our approach to protect gTLD sites, indicating that a semantic model can also be built for some websites under the gTLD domains to capture the promotional attacks on them. Finally, we elaborate on a study on some prominent attack cases discovered in our research, which, from the semantic perspectives, analyzes the techniques the adversary employ in the promotional infections.

6.4.1 Landscape

Scope and magnitude. Our study reveals that the promotional infections are spread across the world, compromising websites in all kinds of sTLDs. Altogether, SEISE detected around 1 million URLs leading to malicious content on 11,473 infected FQDNs under 9,734 sTLD domains. The results are summarized in Table 21 and



Domain	Alexa Rank	IBT	Domain	Alexa Rank	IBT
nih.gov	196	General	purdue.edu	1462	Drug
ca.gov	649	Gambling	cdc.gov	1731	Drug
state.gov	719	Drug	umich.edu	1781	Drug
mit.edu	789	Drug	cornell.edu	1806	Drug
harvard.edu	1034	Cheating	ed.gov	1816	Drug
weather.gov	1035	Software	washington.edu	1905	Drug
stanford.edu	1050	Drug	sp.gov.br	1986	Drug
noaa.gov	1126	Software	ucla.edu	1989	General
psu.edu	1342	Drug	utexas.edu	2012	Financial
berkeley.edu	1452	Drug	wisc.edu	2199	General

Figure 32: Cumulative distribution of injected sTLD sites' Alexa rank and Top 20 injected sTLD sites with highest Alexa rank.

Table 22.

To understand the magnitude of the threat towards individual sTLDs, we studied the ratio of compromised FQDNs under each domain category. For this purpose, we first tried to get some idea about how many FQDNs are under each sTLD, using the passive DNS dataset from DNSDB [51]. The dataset includes the records of individual DNS RRsets as well as first-seen, last-seen timestamps for each domain and the DNS bailiwick from Farsight Security's Security Information Exchange and the authoritative DNS data. The number of FQDNs under an sTLD was estimated from those under the sTLD queried between 2014/01 and 2015/08, as reported by the passive DNS records. The results were further cross-validated by comparing them with the estimated domain counts given by DomainTools [7] for each TLD.

Table 21 illustrates the top-10 sTLD with the largest number of infected domains, together with the number of domains we monitored and the total number of domains we estimated for each sTLD. According to our findings, `gov.cn` is the least protected sTLD with a significant portion of the FQDNs compromised (12%), which is followed by `edu.vn` 3% and `edu.cn` 3%. The top-3 sponsoring registrars with the most infected `gov.cn` sites are `sfn.cn`, `alibaba.com`, `xinnet.com`. On the other hand, `.mil` sites apparently are better protected than others. Among the 456 `.mil` domains we monitored, only 8 domains are injected.

Figure 33 describes the distributions of the compromised sTLD sites across 141

countries, as determined by their geolocation. Based upon the number of infected domains, countries are colored with different shades of blue. As we can see here, most of infected sites are found in China (15%), followed by United States (6%) and Poland (5%).

Impacts of the infections. We further looked into the Alexa ranks of injected sTLD websites, which are presented in Figure 32. Across different sTLDs, highly ranked websites were found to be exploited, getting involved in various types of malicious activities, SEO, Phishing, fake drug selling, academic cheating, etc. Figure 32 illustrates the cumulative distributions of the ranks: a significant portion of the infections (75%) actually happen to those among the top 1M. Figure 32 further shows the top-20 websites with the highest Alexa ranks. Among them, 12 are under `.edu`, including the websites of leading institutions like `mit.edu` (Alexa:789), `harvard.edu` (Alexa:1034), `stanford.edu` (Alexa:1050) and `berkeley.edu` (Alexa:1452), and 7 under `.gov`, such as `nih.gov` (Alexa:196), `state.gov` (Alexa:719) and `noaa.gov` (Alexa:1126). In general, China is the country that hosts most injected sTLD sites; however, when it comes to top ranked sites (Alexa rank $<$ 10K), 67% of them are in the United States and Australia.

Also interesting is the types of malicious activities in which those domains are involved. Table 22 shows the number of the domains utilized for promoting each type of content (across all 10 categories). As we can see here, most of the injected sTLD sites (19%) are in the *Gambling* category, which is followed by those related to *Drug* (15%) and *General Product* (14%) such as shoes and healthcare products. When we look at the top-20 domains, many of them are infected to promote *Drug*. Also, many `.edu` domains advertise unlicensed pharmacy, while `.gov` are mainly compromised to promote gambling and fake AV. Interestingly, the injected domains associated with different countries tend to serve different types of content. For example, the most common promotions on Chinese domains are gambling (which is illegal in that

country), while most injected US domains are linked to unlicensed online pharmacy. Since the infected country code sTLDs (e.g., `.cn`) can make the content they promote more visible to the audience in related countries (e.g., boosting the ranks of malicious sites in the results of country-related searches), it is likely that promotional infections target specific groups of Internet users, just like legitimate advertising.

Our study further shows that many of such infections have been there for a while. Figure 34 shows the distribution of the infection time for the injected page in sTLD sites. We estimated the durations of their infections by continuously crawling the 20K injected pages (which were detected in 2015/08) every two days from 2015/08 to 2015/11 to find out whether they were still alive. As we can see from the figure, most infections last 10-20 days, while some of them have indeed been there for a while, at least 1 months. A prominent example is the injection on `ca.gov`, whose infection starts no later than 60 days.

6.4.2 Implications of Semantics Inconsistency

Our study shows that promotional infections, particularly for those under sTLDs, are characterized by the inconsistency between the semantics of the promoted content and that of an infected domain's generic content: in our labeled bad set (the collection of compromised domains reported by CleanMX; see Section 6.3.2), all sTLD-related infections contain the malicious content inconsistent with the semantics of their hosting websites. The implication of this observation is that by exploiting this feature, a weakness of the sTLD-based promotional infections, a semantic-based approach, like SEISE, can effectively suppress such a threat to sTLDs. This is significant, since our study, as elaborated below, shows that sTLDs are valuable to the adversary because they are less protected and highly ranked. Further, even for gTLDs, which tends to have highly diverse and less specified semantics, the malicious content uploaded there also tends to be incompatible with the compromised websites' themes. This indicates

that our approach can be applied beyond sTLDs. Following we report our findings.

sTLD as a soft target. To understand the importance of sTLDs to the adversary, we compared the compromised sTLD sites with those under the gTLDs, within the same attack campaign. A campaign here includes a set of websites infected for promoting unauthorized or malicious content and those sites share a set of common features, specifically, they all pointing to the same target site being advertised, their malicious URLs having the same features (such as same affiliate ID as URL parameter) and they all share the same redirection chain. In our research, we discovered a campaign through infected websites' "*link-farm*" structure, i.e., a compromised site pointing to another one. Following the links on the compromised sTLD sites enabled us to reach a set of infected gTLD sites, mainly under `.com`. We then compared the features of those sites with those of sTLD domains, in terms of Alexa rank, pagerank (PR) and lifetime, in an attempt to find out what type of TLD domains are more valuable to promotional infections.

Table 23 presents the top-3 campaigns (all organized as link farms) discovered in our study. The largest one covers about 872 sTLDs and 3426 gTLDs across 12 countries and regions (US, China, Taiwan, Hong Kong, Singapore and others). Among the victims are 20 US academic institution such as `nyu.edu`, `ucsd.edu`, 5 government agencies like `va.gov`, `makinghomeaffordable.gov`, together with 188 Chinese universities and 510 Chinese government agencies. Also among the victims are 1507 `.com` sites. Figure 35(a) and Figure 35(b) compare the Alexa global ranks and the page rank (PR) of those gTLD and sTLD websites. As we can see from the figures, 50%-75% of sTLD sites are ranked within the Alexa top 1M, while only 10%-30% of gTLD sites are at this level. Actually, more than 40% of the gTLD sites have Alexa rank outside the top 5M. By comparison, less than 20% of sTLDs have ranks outside the top 5M. In terms of PR, more than 30% of the sTLD sites have PR from 4 to 6, while less than 5% of gTLD sites are PR4-PR6. Also, more than half of gTLD sites

Table 23: Top 3 link-farm campaigns with most injected sTLD domains.

Name	# sTLD domains	# gTLD domains	# countries	Promotion Content
Campaign 1	872	3,426	12	Gambling
Campaign 2	148	5,210	7	Cheating
Campaign 3	60	5,198	15	Drug

have PR as 0, which have a weaker SEO effectiveness than those with high PR. This indicates that the majority of sTLD sites have a stronger effect on the promoted sites than gTLD sites with no or low PR.

We further compared the durations of the infections for these two types of domains. Again, we continuously crawled the compromised pages (identified in 2015/08-2015/09) every two days from 2015/09 to 2015/11 to check whether the infections were still there. Figure 35(c) illustrates the distributions of the sTLD site's life spans and those of gTLD sites. As can be seen from the figure, gTLD sites were cleaned up more quickly than the sTLD sites. Over 25% of the gTLD sites were cleaned within 10 days, while 12% of the sTLD sites were cleaned within 10 days.

Our study demonstrates that the sTLDs are ranked higher than the gTLD sites and much more effective in elevating the ranks of promoted content, thereby more valuable to promotional infections. In the meantime, they are less protected than the gTLDs: once compromised, the infections will stay there for a longer period of time. This indicates that, indeed, the sTLDs are valuable assets to the adversary and effective protection of the site, as SEISE does, indeed makes the promotional attacks less effective.

Extension to gTLDs. Compared with sTLDs, gTLDs (e.g., `.com`, `.net` and `.org`) do not have fixed semantic meanings. However, we found that still the malicious content injected here tends to be incompatible with the semantics of the sites, which can be captured by the search engine results. Figure 36 presents an example of search engine results for an injected gTLD site `iceriversprings.com`, which is the website

of Ice River Green brand of bottled water. However, the injected page show the semantically inconsistent content for "payday loan" promotion.

Then, we measure the semantics inconsistency on the 3,000 gTLD sites, which are randomly sampled from the aforementioned campaigns. Specifically, we use the *Context Analyzer* component in SEISE to calculate the semantic distance between the generic content of those known injected sites (the reference, e.g., the search result of the query `site:iceriversprings.com`) and the results of querying IBTs on these sites, which mostly contain injected malicious content (e.g., `site: iceriversprings.com "payday loan"`). However, we also found that some compromised gTLD sites show semantic consistent with the promotional content. For example, online drug library `druglibrary.org` (in Campaign 3) was injected to promoted "cheap xanax". Hence, to identify those suspicious sites (before they are checked with the *Context Analyzer*), we utilized the similarsites website query API [15] to fetch the site tags (e.g., "recycling" and "water" for `site:iceriversprings.com`) to determine a gTLD site's semantics, and only use the gTLD sites showing semantic inconsistency with the IBT (i.e., the site's tags semantically distance away from the IBT) as the suspicious candidates for the input of the *Context Analyzer*. This filtering step (for the purpose of increasing the "toxicity level" [66] of the inputs) is built as the *Semantic comparator*, which accepts the threshold for the IBT semantics distance (Section 6.2.2) and outputs the candidate gTLD sites that have great semantic distances with the IBT used for the query. For example, `iceriversprings.com`, which has the site tag "recycling", "water" which shows semantic inconsistency (determined by *Semantic comparator* Figure 28) with the IBT "payday loan", will be regarded as suspicious FQDNs and become the input of the *Context Analyzer*.

Figure 35(d) shows the semantic distances between the reference and the search results of querying an IBT with and without the *Semantic comparator*. We observe that the *Context Analyzer* can still identify the semantics inconsistency, particularly

with the help of the *Semantic comparator* that selects sites with great semantic distances with the IBT: 97% of the injected sites have semantic distance larger than 0.8 when the threshold of *Semantic comparator* is set to 0.9; by comparison, 85% of the injected sites have semantic distance larger than 0.8 in the absence of the *Semantic comparator*.

Further, we measure the semantic inconsistency of unknown injected gTLD sites. This is nontrivial because simply searching `site:.com "payday loan"` will return mostly legitimate search results. Even though we could validate these FQDNs one by one through the *Semantic comparator* and the *Context Analyzer*, the cost for finding truly compromised sites becomes overwhelming. As mentioned earlier, with a similar PR, gTLD sites are better protected than sTLD sites. Hence, when searching gTLDs under the IBT (e.g., `site:.com "payday loan"`), high-PR gTLD sites tend to appear on top of the search results, which are actually less likely to be compromised. For example, when searching "payday loan", many high-PR sites such as `checkintocash.com`, `wikipedia.org` and `www.acecashexpress.com` will show up within the top-100 search results. None of them appear to be compromised. To address this challenge and identify the sites likely to be compromised (which will be further determined by the *Context Analyzer*), we utilized long IBTs (word length larger than 4) to feed search engine to obtain suspicious FQDNs. Generally, longer query keywords have less search competition [105], i.e., websites with lower PRs are more likely to appear in the search results. For example, when searching for "payday loan no credit check" under `.com`, bottled water website `iceriversprings.com` and ATM company website `carolinaatm.com` are within the top-10 search results.

In our experiments, we utilized 1000 long IBTs in 10 individual categories to do the search, and 23,098 gTLD FQDNs were collected for the semantic inconsistency analysis. We set the threshold of the *Context Analyzer* to 0.9, and 7,430 of the gTLD FQDNs were reported to have promotional infections. We further randomly

sampled 400 results (200 injected and 200 not-injected) and manually checked the findings. We confirmed that 182 were indeed infections and 196 were not injected, which gives us an FDR of 9% and FPR of 8.4%. With this encouraging outcome, how to detect compromised gTLDs through semantics-based approaches remains to be an open question. Particularly, new techniques need to be developed to further suppress FDR and improve its coverage. Also, query terms for detection should also be automatically discovered.

6.4.3 Case Studies

Perhaps the most surprising findings of our study is the discovery of several large-scale attacks, infecting many leading organizations around the world. In addition to the aforementioned gambling campaign, we also found the infections for promoting counterfeit products, fake essays and political materials on university and government sites. Here we present the studies on two cases as examples to provide additional information about what techniques the adversary uses and how the attacks are organized.

Exploit kit discovered. We found an exploit toolkit used in multiple gambling campaigns, for example, Campaign 1. The toolkit, called `xise`, was discovered on a cloud drive. By analyzing its code, we found that `xise` has the functionalities for automatic site collection, shell acquisition, customized injected page generation and a series of evasion techniques such as redirection cloaking and code obfuscation. More specifically, it automatically discovers the domains of high-profile websites from Google and other search engines, and also scans the websites for the vulnerabilities within the components such as `phpmyadmin`, `kindeditor`, `ueditor`, `alipay` and `fckeditor`. Further, it lets its user provide the promoted site's URL and keywords and automatically generates the pages to be injected to the compromised websites along a specific path (e.g., `filemanager/browser/default/images/icons`). The tool also uploads

Table 24: Example of signatures.

Signature
<!--google1-->...<!--googlee-->   <img width="20" height="20" border="0" hspace="0" vspace="0" src="http: //count51.51yes.com/count1.gif">   <!--ZJEG_RSS.content.begin-->...<!--ZJEG_RSS.content.end-->   <iframe marginwidth="0" marginheight="0" hspace="0" vspace= "0" frameborder="0" scrolling="no" src="" height="0" width="0">

a configuration file to the compromised web server to perform redirection cloaking: i.e., it will redirect visitors based on their HTTP referers to protect the compromised site. Also, to guarantee the malicious content to be indexed by search engines, `xise` also uploads scripts to keep generating pages to guarantee SEO effectiveness. Note that adding and changes is a freshness factor for high search engine ranking. In our research, we manually generated signatures for `xise` as listed in Table 24. 1037 of sTLD sites we detected are related to `xise` with the average semantics distance 0.87 to it sTLDs.

Academic cheating infections. Our research also discovered many infections promoting academic cheating sites. Those sites provide online services for preparing any kind of homework at the high school and college levels, and even taking online tests for students. We found that such attacks mainly aim at `.edu` domains and the examples of the IBTs involved include 'free essay', 'cheap term paper' and others. These terms were found to be very effective at finding such malicious activities. SEISE detected 428 compromised sites, including high-profile `.edu` domains such as `mit.edu`, `princeton.edu`, `havard.edu`, etc.

Table 25 compares the compromised `.edu` sites in different keyword categories. We observe that such malicious activities have apparently already become a global industry. 119 education TLDs in 109 countries have 428 infected domains to promote academic cheating sites. The Top 3 education TLDs with most infected sites are `edu` (23%), `edu.mn` (11%) and `edu.cn` (7%).

Table 25: Comparison of injected education TLDs sites in different keyword categories.

Category	# FQDN	# domains	Performance (term query per site)
Academic cheating	470	428	2.2
Gambling	589	367	2.8
Drug	423	360	2.5
Financial	401	327	3
Adult	260	214	3.2

6.5 Discussion

Our research shows that semantics-inconsistency search offers a highly-effective solution to the promotional-infection threat. In this section, we discuss the tricks the adversary can play to evade our detection, limitations of our technique and future research, together with the lesson learnt from our study and our communication with the victims.

Evasion. The current implementation of SEISE is based upon the search results returned from Google and Bing. While both are mainstream search engines targeted by promotional infections, the data we crawled are limited to the sites that indexed by Google and Bing. Hence, to evade SEISE, the adversary, who has full control of a compromised website, may set `robots.txt` to prevent part of its content from being scanned. Such evasion techniques, however, will cause the promotion pages to lose the visitors from the search engines and also the high-profile links to the sites being promoted. This defeats the purpose of the promotional infections, which are meant to advertise malicious content through the search engines and therefore should aggressively expose its content (promotional pages) to the search engines, instead of hiding it from them. Other issues related to search results include the delay introduced by page indexing and page expiration. Again, although our approach is not designed to capture a promotional infection before it is indexed by the search engines, the

impact of the infection is also limited at that time, simply because its whole purpose is to advertise some malicious materials, which is not well served without the infected pages being discovered by the search engine. For page expiration, we need to consider the fact that as long as the URLs of the promoted content are still alive, the attack is still in effect, since letting people find the URLs is the very purpose of the attack. Whether the URLs are still there can be confirmed by crawling the links. Further, the snippet of the search results, even for the pages that are already expired, can still be utilized to find new keywords.

The adversary may play other evasion tricks, by adding more relevant keywords to the infected page to make the content look more consistent with the website's theme, or hiding the inconsistent content by embedding it within images. However, even in the presence of relevant content, the malicious keywords can still be recovered and cause an observable semantic deviation from the theme of the original website, as long as the keywords are sufficiently frequent to be picked up by the search engine and contribute to the change of the malicious content's rank in search results. Hiding content in images results in neglect of malicious content in the search results, which is not what the adversary wants. Fundamentally, no matter what the adversary does, the fact remains that any attempt to cover the content being advertised will inevitably undermine the effectiveness of the promotional effort. Another evasion strategy is to just compromise the website with compatible semantics. This approach will significantly limit the attack targets the adversary can have. Particularly, it is less clear how this can be done for sTLDs. Note that even selling medicine on a health institution's site can be captured, as the infections of the NIH pages shown at the beginning of the paper.

Limitations. As mentioned earlier, our current design is focused on detecting the infections of sTLD sites, since they have well-defined semantic meanings and are a

soft target for the adversary. In the meantime, gTLDs are also known to be extensively compromised for promotion purposes. A natural follow-up step is to develop the semantic technologies for protecting those domains. This is completely feasible, as demonstrated in our preliminary study (Section 6.4.2): by leveraging the Alexa categories, the semantics of even those more generic domains can also be identified and compared with that of the content it hosts.

Moreover, our semantic-based detection technique does not differentiate between server injected domains, blog/forum Spam and URL redirection [71] (e.g., posting ads on a `.edu` forum or utilizing the server-side script of a `.gov` domain to dynamically create a page under the domain with promotion content, see Section 6.1). In our research, we randomly sampled 100 detected pages and found that about 20% of them are Spam, which are also considered illicit advertising [71]. A follow-up step is to develop automatic technologies to identify those cases, so we can respond to them in a different way (e.g., through input sanitization). For example, a comment page oftentimes can be detected from the keywords such as "comment" or "redirect" involved in its link; such a page, once found to promote malicious content, can be further analyzed to determine whether the content is link Spam or caused by an infection.

Also, the use of search engines has a performance implication. Search service providers often have limits on the crawling frequency one can have, which causes delay in detecting malicious content and affects the scalability of our technique. On the other hand, given the effectiveness of SEISE in catching promotional infections, we believe that a collaboration with the search provider to detect Internet-wide infections is completely possible.

Lesson learnt. Our study shows that sTLD sites are often under-protected. Particularly for universities and other research institutions, their IT infrastructures tend to be open and loosely controlled. As a prominent example, in a university, individual

servers are often protected at the department levels while the university-level IT often only takes care of network-level protection (e.g., intrusion detection). The problem is that, oftentimes, the hosts are administrated by less experienced people and include out-dated and vulnerable software, while given the nature of the promotional infections, they are less conspicuous in the network traffic, compared with other intrusions (e.g., setting up a campus bot net). We believe that SEISE, particularly its Context Analyzer, can play the role of helping the web administrators of these organizations detect the problems with those less-protected hosts. Of course, a more fundamental solution is to have a better centralized control, at least in terms of discovering the security risks at the host level and urging the administrators of these hosts to keep their software up-to-date.

Responsible disclosure. Since the discovery of infected domains, we have been in active communication with the parties affected. So far, we have reported over 120 FQDNs to CERT in US and 136 FQDNs to CCERT (responsible for `.edu.cn`) in China, the two countries hosting most infected domains. By now, CCERT have confirmed our report, and notified all related organizations, in which 27 responded and fixed their problems. However, it is difficult for us to directly contact the victims to get more details (like log access) from the infected servers. On the other hand, given the scale of the attacks we discovered, the whole reporting process will take time.

6.6 Summary

In this chapter, we report our study on promotional infections, which introduce a large semantic gap between the infected sTLD and the illicit promotional content injected. Exploiting this gap, our semantic-based approach, SEISE, utilizes NLP techniques to automatically choose IBTs and analyze search result pages to find those truly compromised. Our study shows that SEISE introduces low false detection rate (about

1.5%) with over 90% coverage. It is also capable of automatically expanding its IBT list to not only include new terms but also terms from new IBT categories. Running on 100K FQDNs, SEISE automatically detects 11K infected FQDN, which brings to light the significant impact of the promotional infections: among those infected are the domains belonging to leading educational institutions, government agencies, even the military, with 3% of `.edu` and `.gov`, and over one thousand domains of `.gov.cn` falling prey to illicit advertising campaigns. Our research further demonstrates the importance of sTLDs to the adversary and the bar our technique raises for the attacks. Moving forward, we believe that there is a great potential to extend the technique for protecting gTLDs, as indicated by our preliminary study. Further, we are exploring the possibility to provide a public service for detecting such infections.

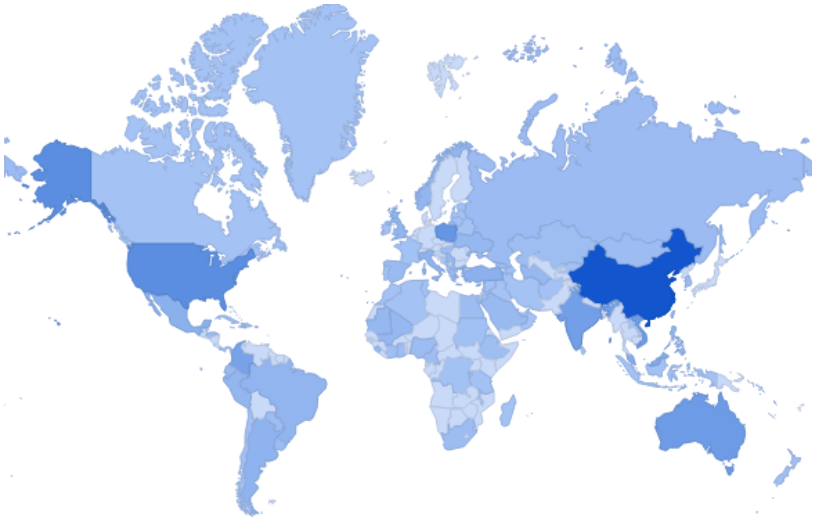


Figure 33: The geolocation distributions of the compromised sTLD sites across 141 countries.

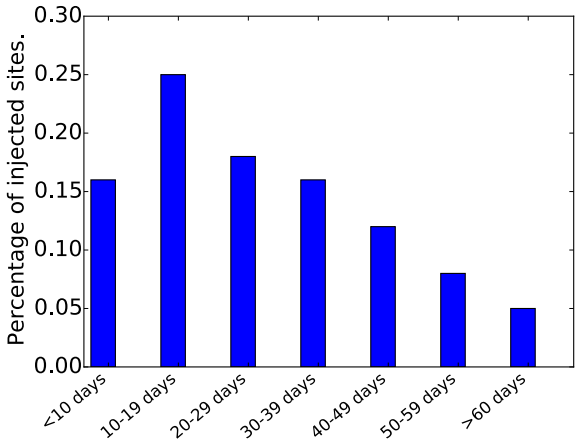
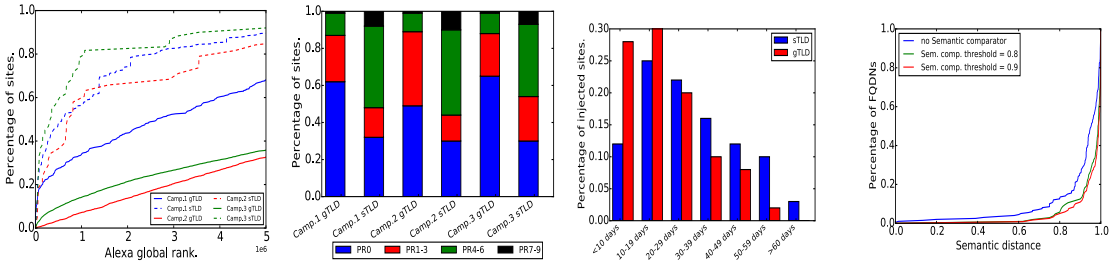


Figure 34: The distribution of the infection time.



(a) **Cumulative distribution of Alexa global ranks per sites in 3 campaigns.**

(b) **Cumulative distribution of Alexa bounce rate per sites in 3 campaigns.**

(c) **Distribution of the infection time for the injected pages in sTLD sites and gTLD sites.**

(d) **Cumulative distribution of semantics distance per monitored sites.**

Figure 35: Alexa global rank, PR and life span of sites in three campaigns, and cumulative distribution of semantics distance per monitored sites.

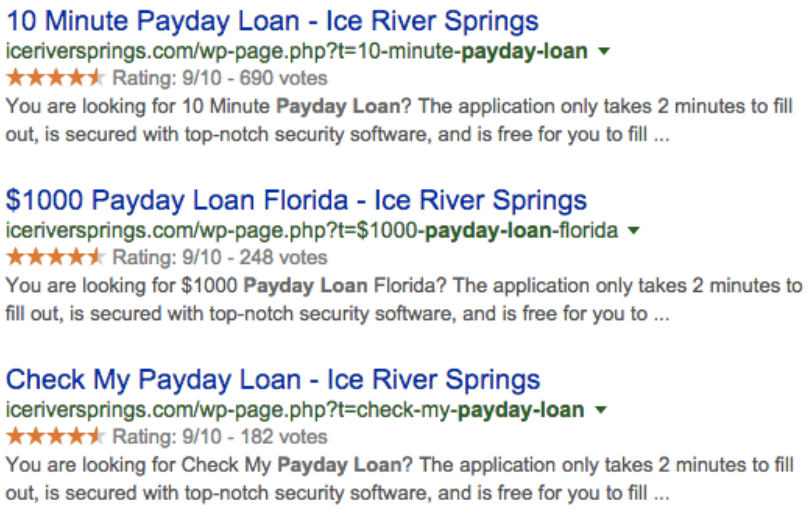


Figure 36: Example of search engine results of an injected gTLD site iceriver-springs.com.

CHAPTER VII

CONCLUSION

The cyber threat landscape is quickly changing, and it is of vital importance to stay abreast of emerging threats and to proactively work to improve security. At the same time, piecing together a complete landscape of attacks by identifying the strategies and capabilities of the adversaries requires establishing links among individual observations. Also, defending against these attacks requires automatically generated semantics-aware policies to complement manual analysis. While using data processing techniques and semantic-aware inspection to address security problems is a promising approach to evaluate security risks and to provide cyber intelligence, there exists a gap between the security ontology and general data processing techniques primitives needed for such an approach. This difference tends to be domain-sensitive, language-specific, and computationally intensive which further complicates the use of such an approach.

In this dissertation, data processing techniques and semantic-aware inspection were applied in three components of the modern security system: cyber threat gathering, cyber threat analysis, and cyber threat detection. From the developed modern security system, three techniques are presented as contributions: (1) we present an innovation solution for fully automated IOC extraction. It solved a name entity recognition problem for security domain through an innovative graph mining technique. Running on 71,000 articles collected from 45 leading technical blogs, this new approach demonstrated a remarkable performance: it generated 900K OpenIOC items with a precision of 95% and a coverage over 90%, which is way beyond what the state-of-the-art NLP technique and industry IOC tool can achieve, at a speed of thousands

of articles per hour. Further, by correlating the IOCs mined from the articles published over a 13-year span, our study shed new light on the links across hundreds of seemingly unrelated attack instances, particularly their shared infrastructure resources, as well as the impacts of such open-source threat intelligence on security protection and evolution of attack strategies. (2) we analyzed emerging cyber threats on cloud platform using big data analytics technique. We identified a set of collective features, which uniquely characterize malicious cloud repositories. These features were utilized to build a scanner that detected over 600 malicious cloud repositories on leading cloud platforms like Amazon, Google, and 150K sites, including popular ones like `groupon.com`, using them. Highlights of our study include the pivotal roles played by these repositories on malicious infrastructures, and other important discoveries include how the adversary exploited legitimate cloud repositories and why the adversary uses Bars in the first place that had never been reported. These findings bring such malicious services to the spotlight and contribute to a better understanding and ultimately eliminating this new threat. (3) we developed a semantic-based technique, called *Semantic Inconsistency Search* (SEISE), for efficient and accurate detection of the promotional injections on sponsored top-level domains (sTLD) with explicit semantic meanings. Running on 403 sTLDs with an initial 30 seed IBTs, SEISE analyzed 100K fully qualified domain names (FQDN), and along the way automatically gathered nearly 600 IBTs. In the end, our approach detected 11K infected FQDN with a false detection rate of 1.5% and over 90% coverage. Our findings further bring to light the stunning impacts of such promotional attacks, which compromise FQDNs under 3% of `.edu`, `.gov` domains and over one thousand `gov.cn` domains, including those of leading universities such as `stanford.edu`, `mit.edu`, `princeton.edu`, `havard.edu` and government institutes such as `nsf.gov` and `nih.gov`. We further demonstrate the potential to extend our current technique to protect generic domains such as `.com` and `.org`.

REFERENCES

[1] "A community OpenIOC resource." https://openiocdb.com/.

[2] "Beautiful Soup." https://www.crummy.com/software/BeautifulSoup/.

[3] "Bing search api.." https://datamarket.azure.com/dataset/bing/search.

[4] "Buckets." https://cloud.google.com/storage/docs/json_api/v1/buckets. [Online].

[5] "Common Vulnerabilities and Exposures." https://cve.mitre.org/.

[6] "Defense Industrial Base Cybersecurity Information Sharing Program." http://dibnet.dod.mil/.

[7] "Domaintools." https://www.domaintools.com.

[8] "GIMP." https://www.gimp.org/downloads/.

[9] "Google groups." https://groups.google.com/.

[10] "Google web search api.." https://developers.google.com/web-search/?hl=en.

[11] "Open Source OCR Engine." https://github.com/tesseract-ocr/tesseract.

[12] "Phishtank." https://www.phishtank.com.

[13] "SecurityFocus." www.securityfocus.com/.

[14] "Servnet." https://servnetshsztndci.onion. [Online].

[15] "Similar websites api." https://developer.similarweb.com/similar_websites_api.

[16] "Stopword lists." http://www.ranks.nl/stopwords.

[17] "Structured Threat Information eXpression." https://stixproject.github.io.

[18] "The OpenIOC Framework." http://www.openioc.org.

[19] "topia.termextract 1.1.0." https://pypi.python.org/pypi/topia.termextract.

[20] "Virustotal." https://www.virustotal.com/.

[21] "word2vec, tool for computing continuous distributed representations of words.." https://code.google.com/p/word2vec/.

[22] "YARA.." http://plusvic.github.io/yara/.

[23] "Words and phrases that trigger some spam filters." http://webmarketingtoday.com/articles/spamfilter_phrases/, 2002.

[24] "Email spam filter trigger words to avoid in your e-campaigns." http://www.mannixmarketing.com/blog/spam-trigger-words/, 2009.

[25] "50 of the most competitive seo keywords!." https://moz.com/ugc/50-of-the-most-competitive-seo-keywords, 2012.

[26] "Real-Time Threat Intelligence." https://www.recordedfuture.com/, 2016.

[27] "Semantic Link: find related words." http://semantic-link.com/, 2016.

[28] Abiword, "Enchant." http://www.abisource.com/projects/enchant/, 2010.

[29] Alias-i, "Lingpipe 4.1.0." http://alias-i.com/lingpipe, 2008.

[30] AlienVault, "Open Threat Intelligence." https://otx.alienvault.com/, 2016.

[31] Alrwais, S., Yuan, K., Alowaisheq, E., Li, Z., and Wang, X., "Understanding the dark side of domain parking," in *Proceedings of the 23rd USENIX Security Symposium*, 2014.

[32] Bach, N. and Badaskar, S., "A review of relation extraction," *Literature review for Language and Statistics II*, 2007.

[33] Bishop, C. M., *Pattern recognition and machine learning*. springer, 2006.

[34] Borgolte, K., Kruegel, C., and Vigna, G., "Delta: automatic identification of unknown web-based infection campaigns," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pp. 109–120, ACM, 2013.

[35] Borgolte, K., Kruegel, C., and Vigna, G., "Meerkat: detecting website defacements through image-based object recognition," in *Proceedings of the 24th USENIX Conference on Security Symposium*, pp. 595–610, USENIX Association, 2015.

[36] BuiltWith, "Builtwith." http://builtwith.com/, 2015. [Online].

[37] Caballero, J., Grier, C., Kreibich, C., and Paxson, V., "Measuring Pay-per-Install: The Commoditization of Malware Distribution," in *Proc. 20th USENIX Security Symposium*, (San Francisco, CA), Aug. 2011.

[38] Catakoglu, O., Balduzzi, M., and Balzarotti, D., "Automatic extraction of indicators of compromise for web applications," in *WWW 2016*, 2016.

[39] Chen, D. and Manning, C. D., "A fast and accurate dependency parser using neural networks.," in *EMNLP*, pp. 740–750, 2014.

[40] Chris Kanich and Christian Kreibich and Kirill Levchenko and Brandon Enright and Vern Paxson and Geoffrey M. Voelker and Stefan Savage,, "Spamalytics: an Empirical Analysis of Spam Marketing Conversion," in *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, (Arlington, VA), Oct. 2008.

[41] Clarke, J., Srikumar, V., Sammons, M., and Roth, D., "An nlp curator (or: How i learned to stop worrying and love nlp pipelines)," in *LREC*, 5 2012.

[42] Clean-MX, "Clean mx realtime database." `http://support.clean-mx.de/clean-mx/viruses.php`, 2015. [Online].

[43] Cohen, W., Ravikumar, P., and Fienberg, S., "A comparison of string metrics for matching names and records," in *Proceedings of Kdd workshop on data cleaning and object consolidation*, 2003.

[44] Comm100, "Spammy words." `http://emailmarketing.comm100.com/email-marketing-ebook/spam-words.aspx`, 2015. [Online].

[45] Cook, D. J. and Holder, L. B., *Mining graph data*. John Wiley & Sons, 2006.

[46] Crawl, C., "Common crawl." `https://commoncrawl.org/`, 2015. [Online].

[47] Culotta, A. and Sorensen, J., "Dependency tree kernels for relation extraction," in *Proceedings of ACL'04*.

[48] damballa, "Dgas in the hands of cyber-criminals:examining the state of the art in malware evasion techniques." `https://www.damballa.com/downloads/r_pubs/WP_DGAs-in-the-Hands-of-Cyber-Criminals.pdf`, 2015. [Online].

[49] Der, M. F., Saul, L. K., Savage, S., and Voelker, G. M., "Knock it off: Profiling the online storefronts of counterfeit merchandise," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1759–1768, ACM, 2014.

[50] Der, M. F., Saul, L. K., Savage, S., and Voelker, G. M., "Knock it off: profiling the online storefronts of counterfeit merchandise," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1759–1768, ACM, 2014.

[51] DNSDB, "Passivedns." `https://www.dnsdb.info/`, 2015. [Online].

[52] Facebook. `https://developers.facebook.com/products/threat-exchange`.

[53] Finkel, J. R., Grenager, T., and others, "Incorporating non-local information into information extraction systems by gibbs sampling," in *Proceedings of ACL'05*.

[54] Gärtner, T., Flach, P., and Wrobel, S., "On graph kernels: Hardness results and efficient alternatives," in *Learning Theory and Kernel Machines*, 2003.

[55] Google, "Google Trend." `http://www.google.com/trends/hottrends`, 2014. [Online].

[56] Google, "Rich snippets guidelines." https://support.google.com/webmasters/answer/2722261?hl=en, 2014. [Online].

[57] Google, "Webmaster Guidelines." https://support.google.com/webmasters/answer/35769?hl=en&ref_topic=6002025, 2014. [Online].

[58] Google, "Google hosted libraries." https://developers.google.com/speed/libraries/?csw=1, 2015. [Online].

[59] Google, "Publish website content." https://developers.google.com/drive/web/publish-site, 2015. [Online].

[60] Grier, C., Ballard, L., Caballero, J., Chachra, N., Dietrich, C. J., Levchenko, K., Mavrommatis, P., McCoy, D., Nappa, A., Pitsillidis, A., and others, "Manufacturing compromise: the emergence of exploit-as-a-service," in *Proceedings of the 2012 ACM conference on Computer and communications security*, pp. 821–832, ACM, 2012.

[61] Gusfield, D., "Efficient algorithms for inferring evolutionary trees," *Networks*, vol. 21, no. 1, pp. 19–28, 1991.

[62] Gyongyi, Z., Garcia-Molina, H., and Pedersen, J., "Combating Web Spam with TrustRank," in *Proc. 30th International Conference on Very Large Data Bases (VLDB)*, (Toronto, Canada), Sept. 2004.

[63] Hao, S., Feamster, N., and Pandrangi, R., "Monitoring the initial dns behavior of malicious domains," in *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pp. 269–278, ACM, 2011.

[64] Health, I., "IMS Health." http://www.imshealth.com/portal/site/imshealth, 2014. [Online].

[65] Huang, J., Li, Z., and Xiao, X., "Supor: Precise and scalable sensitive user input detection for android apps," in *USENIX Security'15*.

[66] Invernizzi, L., Comparetti, P. M., Benvenuti, S., Kruegel, C., Cova, M., and Vigna, G., "Evilseed: A guided approach to finding malicious web pages," in *Security and Privacy (SP), 2012 IEEE Symposium on*, pp. 428–442, IEEE, 2012.

[67] IOCbucket, "IOCbucket." https://www.iocbucket.com/, 2016.

[68] John, J. P., Yu, F., Xie, Y., Krishnamurthy, A., and Abadi, M., "deSEO: Combating Search-Result Poisoning," in *Proc. 20th USENIX Security Symposium*, (San Francisco, CA), Aug. 2011.

[69] Josh Grunzweig, "Alina: Casting a Shadow on POS." https://www.trustwave.com/Resources/SpiderLabs-Blog/Alina--Casting-a-Shadow-on-POS/, 2013.

[70] Langville, A. N. and Meyer, C. D., *Google's PageRank and beyond: The science of search engine rankings*. Princeton University Press, 2011.

[71] Leontiadis, N., Moore, T., and Christin, N., "Measuring and analyzing search-redirection attacks in the illicit online prescription drug trade.," in *USENIX Security Symposium*, 2011.

[72] Leontiadis, N., Moore, T., and Christin, N., "Measuring and Analyzing Search-Redirection Attacks in the Illicit Online Prescription Drug Trade," in *Proc. 20th USENIX Security Symposium*, (San Francisco, CA), Aug. 2011.

[73] Leontiadis, N., Moore, T., and Christin, N., "A Nearly Four-Year Longitudinal Study of Search-Engine Poisoning," in *Proc. 21st Conference on Computer and Communications Security (CCS)*, (Scottsdale, AZ), Oct. 2014.

[74] Leontiadis, N., Moore, T., and Christin, N., "A nearly four-year longitudinal study of search-engine poisoning," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pp. 930–941, ACM, 2014.

[75] Levchenko, K., Chachra, N., Enright, B., Felegyhazi, M., Grier, C., Halvorson, T., Kanich, C., Kreibich, C., Liu, H., McCoy, D., Pitsillidis, A., Weaver, N., Paxson, V., Voelker, G. M., and Savage, S., "Click Trajectories: End-to-End Analysis of the Spam Value Chain," in *Proc. IEEE Symposium on Security and Privacy*, (Oakland, CA), May 2011.

[76] Lew, A. A., "Long tail tourism: New geographies for marketing niche tourism products," *Journal of Travel & Tourism Marketing*, vol. 25, no. 3-4, pp. 409–419, 2008.

[77] Li, Z., Alrwais, S., Wang, X., and Alowaisheq, E., "Hunting the red fox online: Understanding and detection of mass redirect-script injections," in *Security and Privacy (SP), 2014 IEEE Symposium on*, pp. 3–18, IEEE, 2014.

[78] Li, Z., Alrwais, S., Wang, X., and Alowaisheq, E., "Hunting the red fox online: Understanding and detection of mass redirect-script injections," in *Security and Privacy (SP), 2014 IEEE Symposium on*, pp. 3–18, IEEE, 2014.

[79] Li, Z., Alrwais, S., Xie, Y., Yu, F., and Wang, X., "Finding the linchpins of the dark web: a study on topologically dedicated hosts on malicious web infrastructures," in *Security and Privacy (SP), 2013 IEEE Symposium on*, pp. 112–126, IEEE, 2013.

[80] Liao, X., Yuan, K., Wang, X., and others, "Seeking nonsense, looking for trouble: Efficient promotional-infection detection through semantic inconsistency search," in *Proceedings of S&P'16*.

[81] Liao, X., Yuan, K., Wang, X., Li, Z., Xing, L., and Beyah, R., "Acing the ioc game: Toward automatic discovery and analysis of open-source cyber threat intelligence," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 755–766, ACM, 2016.

[82] McCoy, D., Pitsillidis, A., Jordan, G., Weaver, N., Kreibich, C., Krebs, B., Voelker, G. M., Savage, S., and Levchenko, K., "PharmaLeaks: Understanding the Business of Online Pharmaceutical Affiliate Programs," in *Proc. 21st USENIX Security Symposium*, (Bellevue, WA), Aug. 2012.

[83] Mihalcea, R. and Tarau, P., "Textrank: Bringing order into texts," Association for Computational Linguistics, 2004.

[84] Mitzenmacher, M. and Upfal, E., *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005.

[85] Moore, T., Leontiadis, N., and Christin, N., "Fashion crimes: trending-term exploitation on the web," in *Proceedings of the 18th ACM conference on Computer and communications security*, pp. 455–466, ACM, 2011.

[86] Moore, T., Leontiadis, N., and Christin, N., "Fashion crimes: trending-term exploitation on the web," in *Proceedings of the 18th ACM conference on Computer and communications security*, pp. 455–466, ACM, 2011.

[87] Nadeau, D. and Sekine, S., "A survey of named entity recognition and classification," *Lingvisticae Investigationes*, vol. 30, no. 1, pp. 3–26, 2007.

[88] Nan, Y., Yang, M., Yang, Z., and others, "Uipicker: User-input privacy identification in mobile applications," in *USENIX Security'15*.

[89] Nelms, T., Perdisci, R., Antonakakis, M., and Ahamad, M., "Webwitness: Investigating, categorizing, and mitigating malware download paths," in *24th USENIX Security Symposium (USENIX Security 15)*, (Washington, D.C.), pp. 1025–1040, USENIX Association, Aug. 2015.

[90] Ntoulas, A., Najork, M., Manasse, M., and Fetterly, D., "Detecting Spam Web Pages through Content Analysis," in *Proc. 15th International Wordl Wide Web Conference (WWW)*, (Edinburgh, Scotland), May 2006.

[91] Obrst, L., Chase, P., and Markeloff, R., "Developing an ontology of the cyber security domain.," in *STIDS*, pp. 49–56, 2012.

[92] Pandita, R., Xiao, X., and others, "Whyper: Towards automating risk assessment of mobile applications," in *USENIX Security'13*.

[93] Platt, J. C., "Sequential minimal optimization: A fast algorithm for training support vector machines," tech. rep., 1998.

[94] Qu, Z., Rastogi, V., and Zhang, X., "Autocog: Measuring the description-to-permission fidelity in android applications," CCS '14.

[95] Ramon, J. and Gärtner, T., "Expressivity versus efficiency of graph kernels," in *First international workshop on mining graphs, trees and sequences*, pp. 65–74, Citeseer, 2003.

[96] Recorded Future. `http://info.recordedfuture.com/Portals/252628/resources/cyber-anatomy-white-paper.pdf`.

[97] Recorded Future, "Recorded Future at SITA." `https://go.recordedfuture.com/hs-fs/hub/252628/file-2607572540-pdf/case-studies/sita.pdf`, 2015.

[98] Reviewopedia, "Reviewopedia." `http://www.reviewopedia.com/`, 2015. [Online].

[99] Rob McMillan, "Open Threat Intelligence." https://www.gartner.com/doc/2487216/definition-threat-intelligence, 2013.

[100] Sabottke, C., Suciu, O., and others, "Vulnerability disclosure in the age of social media: Exploiting twitter for predicting real-world exploits," in *USENIX Security'15*.

[101] Samosseiko, D., "The Partnerka What Is It, and Why Should You Care? ," in *Proc. of Virus Bulletin Conference*, (Geneva, Switzerland), Sept. 2009.

[102] Schmitz, M., Bart, R., Soderland, S., and others, "Open language learning for information extraction," in *Proceedings of the JCEMNLP'12*.

[103] Scipy, "scipy.cluster.hierarchy.linkage." http://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.linkage.html, 2015. [Online].

[104] Settles, B., "Abner: an open source tool for automatically tagging genes, proteins and other entity names in text," *Bioinformatics*, vol. 21, no. 14, pp. 3191–3192, 2005.

[105] Skiera, B., Eckert, J., and Hinz, O., "An analysis of the importance of the long tail in search engine marketing," *Electronic Commerce Research and Applications*, vol. 9, no. 6, pp. 488–494, 2010.

[106] Sklearn, "sklearn.svm.svc." http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html, 2015. [Online].

[107] Snort, "Snort ssl and tls." http://manual.snort.org/node147.html, 2015. [Online].

[108] solutionary, "Threat-intelligence." https://www.solutionary.com/_assets/pdf/research/sert-q4-2013-threat-intelligence.pdf, 2015. [Online].

[109] Sophos, "Security threat report, mid-year 2011." https://www.sophos.com/en-us/medialibrary/Gated%20Assets/white%20papers/sophossecuritythreatreportmidyear2011wpna.pdf, 2011.

[110] Soska, K. and Christin, N., "Automatically detecting vulnerable websites before they turn malicious," in *Proc. USENIX Security*, 2014.

[111] Stringhini, G., Kruegel, C., and Vigna, G., "Shady paths: Leveraging surfing crowds to detect malicious web pages," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pp. 133–144, ACM, 2013.

[112] Sucuri, "Sucuri." https://sucuri.net/, 2015. [Online].

[113] Symantec, "The future of ids." http://www.symantec.com/connect/articles/future-ids, 2015. [Online].

[114] Urvoy, T., Chauveau, E., Filoche, P., and Lavergne, T., "Tracking Web Spam with HTML Style Similarities.," *ACM Transactions on the Web*, vol. 2, no. 1, 2008.

[115] VirusTotal, "Virustotal." https://www.virustotal.com/, 2015. [Online].

[116] Wadleigh, J., Drew, J., and Moore, T., "The e-commerce market for lemons: Identification and analysis of websites selling counterfeit goods," in *Proceedings of the 24th International Conference on World Wide Web*, pp. 1188–1197, International World Wide Web Conferences Steering Committee, 2015.

[117] Wang, D. Y., Der, M., Karami, M., Saul, L., McCoy, D., Savage, S., and Voelker, G. M., "Search+ seizure: The effectiveness of interventions on seo campaigns," in *Proceedings of the 2014 Conference on Internet Measurement Conference*, pp. 359–372, ACM, 2014.

[118] Wang, D. Y., Savage, S., and Voelker, G. M., "Cloak and dagger: dynamics of web search cloaking," in *Proceedings of the 18th ACM conference on Computer and communications security*, pp. 477–490, ACM, 2011.

[119] Wang, D. Y., Savage, S., and Voelker, G. M., "Juice: A longitudinal study of an seo botnet.," in *NDSS*, 2013.

[120] WhatWeb, "Whatweb." `http://www.morningstarsecurity.com/research/whatweb`, 2015. [Online].

[121] Wu, B. and Davison, B. D., "Identifying Link Farm Spam Pages," in *Proc. 14th International Wordl Wide Web Conference (WWW)*, (Chiba, Japan), May 2005.

[122] Wu, F. and Weld, D. S., "Open information extraction using wikipedia," in *Proceedings of ACL'10*.

[123] Xue, N. and others, "Chinese word segmentation as character tagging," *Computational Linguistics and Chinese Language Processing*, vol. 8, no. 1, pp. 29–48, 2003.

[124] Yahoo, "Yahoo! Content Analysis API." `https://developer.yahoo.com/contentanalysis`, 2015. [Online].

[125] Yi, L., Liu, B., and Li, X., "Eliminating noisy information in web pages for data mining," in *Proceedings of ACM SIGKDD'03*.

[126] Yu, L., Zhang, T., Luo, X., and Xue, L., "Autoppg: Towards automatic generation of privacy policy for android applications," SPSM '15, 2015.

[127] Zimmeck, S. and Bellovin, S. M., "Privee: An architecture for automatically analyzing web privacy policies," in *USENIX Security 14*, 2014.